HAGA®

KD9

KD9P universal multi-channel PID compact-controller and SP programmer

User's Manual

HAGA Automation Ltd
1037 Budapest
Királylaki út 35.
T/F 36 1 368-2255 and 36 1 368-6002
E-mail: haga@t-online.hu
Website: www.hagamat.hu

v 1.08
ME190-15

1. Kd9 controller

v 1.08

1.1. Introduction

You can solve complex control tasks with the **1/8 DIN** format **KD9** controller. This controller performs reliable in industrial and laboratory environment, enclosed in an **IP67** protected Polycarbonate housing. The controller contains the properties of a **discrete action controller** (ON/OFF, P, PD PI, PID), **SP controller**, a **setpoint programmer**, a **mini PLC** and a **sequencer controller**.

Due to its many inputs and outputs can be used instead of more simple controllers. The wiring will be more simple and well arranged. There are logic functions among the ALARM-s so you can use a "software screwdriver for relays" to wire the relay sequence by configuration.

There are 7 customized **linearization tables with one variable** and 3 customized **linearization tables with two variables** in the controller in user made table form. The program interpolates for the arguments in the table.

The robust **PID** algorithm contains expert parts, which mend the fast obtaining the setpoint with minimal overshot, and eliminates the sensibility to transients. The **autotune** algorithm gives good parameters for control.

There are 16 PID parameter sets for systems with changing properties in every control loop. The "**Gain scheduling**" sets the proper PID parameters on the whole control range.

The 16 **ALARM** functions are configurable in the same way (without exceptions). There are 270.10^9 variations in every ALARM function so you can hardly meet an insoluble problem. You can connect time relay with latch to the ALARM function.

The setpoint programmer has a lot's of special properties. 4 setpoints are programmable on the same time-base. During running the setpoint program all of the other control functions are working. The setpoint program contains some **HAGA-BASIC** commands too.

The controller can change data with other instruments through its **digital inputs and outputs**.

The data of the controlled system can be send out by the built in **printer interface** and the built in **RS485, RS232 interface** and the built in **M**ulti **M**edia **C**ard inteface. The built in **printer interface** drives the matrix printer through a parallel cable. The RS485, RS232 interface uses **MODBUS** protocol. The addressees and codes are listed in the paragraph of this manual: MODBUS Register.

The Built-in **MMC** memory interface provides a real-time data logging.

Thank to the extraordinary capability of the controller it can control very large systems. After configuring the controller works very safe and reliable. If any disorder occurs it sends special message about the mistake. The level and action of an **error message** are configurable. The KD9 has a very effective **multi-level protection** for the system security.

There are **query functions** in the controller. The KD9 has many signals on the bezel, but cannot show all, what is happening inside it. Therefore, the query informs you about the necessary information. The **Stnd.** is the page for the query. Another method is for the query the ALARM functions. You can configure an ALARM to the required information and when the information will be valid, the signal may be appear on the bezel.

1.2. About this Manual

It is obvious that the KD9 can carry out complex controller tasks. The Manual is essential for commissioning and using.

Before commissioning, please study the structure and contents of the Manual. We propose that connect the controller to the mains. Couple outer LED-s through proper power-supply voltage (battery) to the relays. Couple potentiometers to the inputs. Try to configure this virtual system. Study the controller step by step according to the Manual.

The configuration process is very logic. Figures, flowcharts, tables are helping you during the configuration. The operation method is divided up blocks, groups by the software of the controller. All of the blocks and groups are equal for the same activity e.g. all of the control blocks (PID) are the same except its ordinals.

The controller is handled by the keys on front panel or by a computer with the communication software. Please learn this in the section Handling.

There is not a uniform terminology for these types of controllers. Therefore, we had to use our own. It is very difficult to make an accurate and easy to understand terminology. There are lots of confusing things in the defining the SP programmer configuration menu items, name of instructions and principles of actuation. So, we will use the ones below:

SP programmer: an independent block, which runs from a start to an end (from On to OFF). The Sp programmer saves instructions in its memory, which is divided in steps (segments). These are called program **step** or **segment**. The SP program steps (segments) are saved in memory cells, which are numbered from 00.00 to 99.99. The firs two number defines an SP program, called **profile** (**program**). The second two numbers defines a **step** (**segment**). E.g. 12.34 is the 34th. step (segment) of the 12th. profile (program).

So, we use consequently: write an SP program, or a program or a profile (not configuring program)
part of the profile is a program step, or step, or program segment, or segment
configuring the controller is a process for set the properties of actuating (not programming)

The inputs, blocks, channels, outputs, etc. are the parts of the algorithm. The control loops are build up by these elements. Therefore, when we speak about a control loop we use these names:

PID channel is a control loop from the input to the output

Control channel is a control loop which actuates its output by an ALARM

Control block determines the properties of PV, SP, PID, Y

Input is assembled of physical inputs and other things

Output the result of computation of control data

INP* are six physical inputs on GND, 1, ...6 pins

1.3. Installation and, wiring

Please find the components for installation in the accessory bag.

Cut out: $92^{+0,8}$ x $45^{+0,6}$ for protection IP67 proposed: $92^{+0,4}$ x $45^{+0,3}$

The connectors are at the backside arranged in three levels. Please be careful when connecting the MAINS. The numbers of the connectors are white in black field. In this circuit a T315 mA fuse is necessary. Connect up fuses to the other circuits with the appropriate value.

The disposition of connectors can be seen on Figure 1.

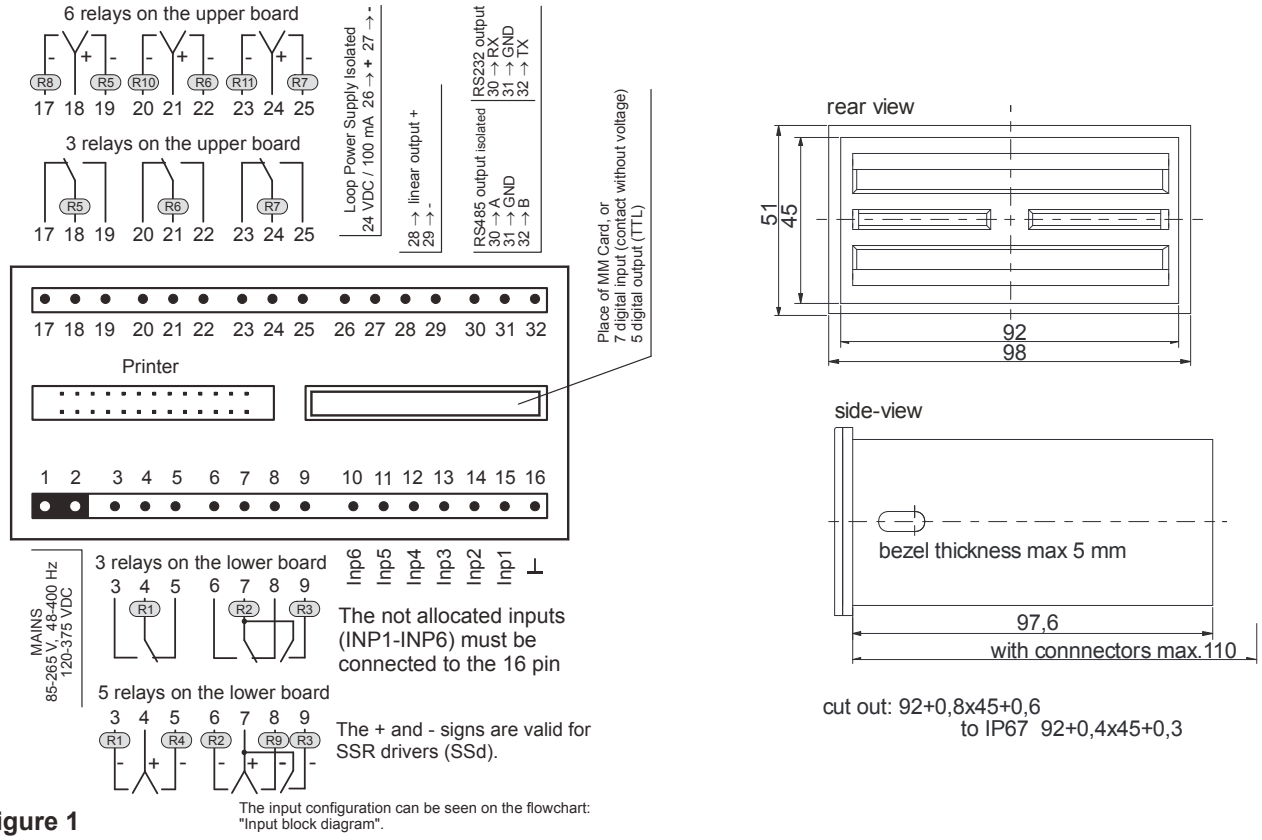

6 relays on the upper board
R8 R5 R10 R6 R11 R7
17 18 19 20 21 22 23 24 25
3 relays on the upper board
R5 R6 R7
17 18 19 20 21 22 23 24 25
Loop Power Supply Isolated
24 VDC / 100 mA 26 → + 27 → -
28 → linear output +
29 → -
RS232 output
30 → RX
31 → GND
32 → TX
RS485 output isolated
30 → A
31 → GND
32 → B
Place of MM Card, or
7 digital input (contact without voltage)
5 digital output (TTL)
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Printer
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
MAINS
85-265 V, 48-400 Hz
120-375 VDC
3 relays on the lower board
3 4 5 6 7 8 9
R1 R2 R3
Inp6 Inp5 Inp4 Inp3 Inp2 Inp1 ⊥
The not allocated inputs
(INP1-INP6) must be
connected to the 16 pin
5 relays on the lower board
3 4 5 6 7 8 9
R1 R4 R2 R9 R3
The + and - signs are valid for
SSR drivers (SSd).
rear view
51
45
92
98
side-view
bezel thickness max 5 mm
97,6
with connnectors max.110
cut out: 92+0,8x45+0,6
to IP67 92+0,4x45+0,3

Figure 1

The input configuration can be seen on the flowchart: "Input block diagram".

In place of relays can be OPC (open collector, 12 V / 20 mA) output for SSR (SSd=SSR driver)

The galvanic isolated pins are: relays, RS485, linear outputs

The galvanic not isolated pins are: analogue inputs, digital inputs, digital outputs, SSd-s, printer interface, RS232 output.

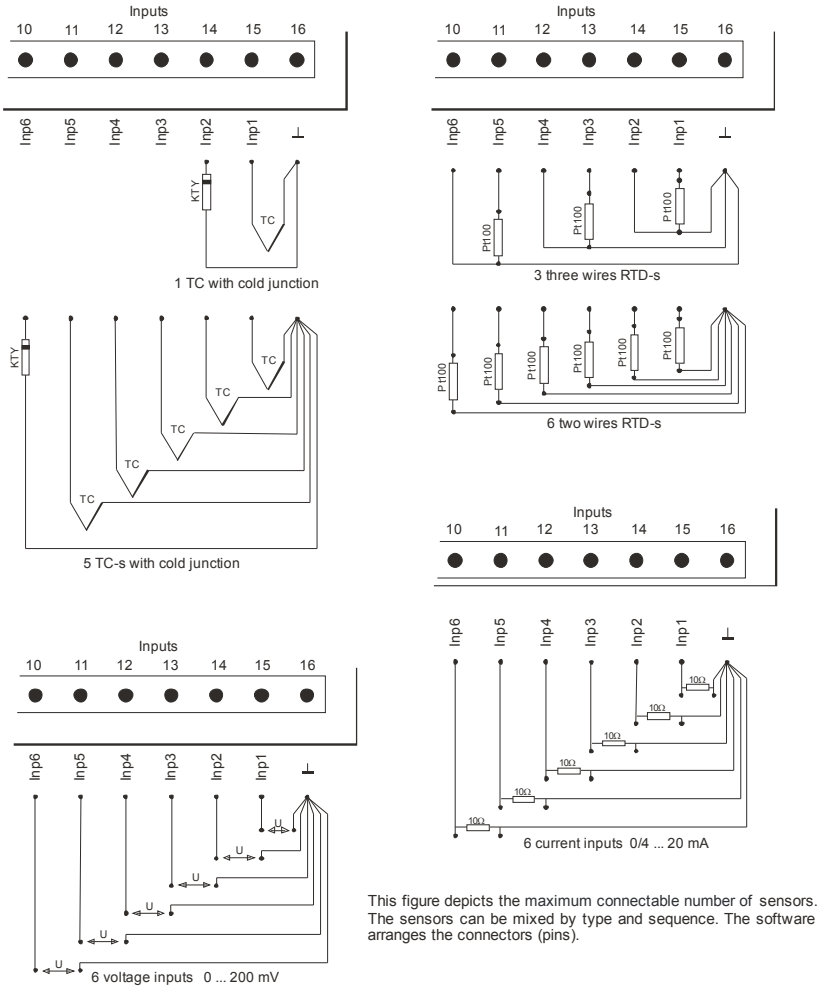
Please be careful when commissioning the controller. The MAINS may cause damage if connected to these pins.

The wiring up of sensors can be seen on Figure 2. You could choose sensor from the table **CAL./In*/dEF.** Another possibility to write the characteristic of the unknown sensor in the **utb*** or **mtr*.** page (customized linearization tables).

Wire up the cold-junction sensor to INP2 when using 1 sensor and to INP6 in case more sensors usage (for TC only).

The sensors occupy their places by the software automatically when configuring. Depending on the number of wire they occupy two or three places (pins) each. E. g.: if the first sensor has three wires it occupies the INP1 and INP2 pins. So the next can be placed to the INP3 and INP4 etc.

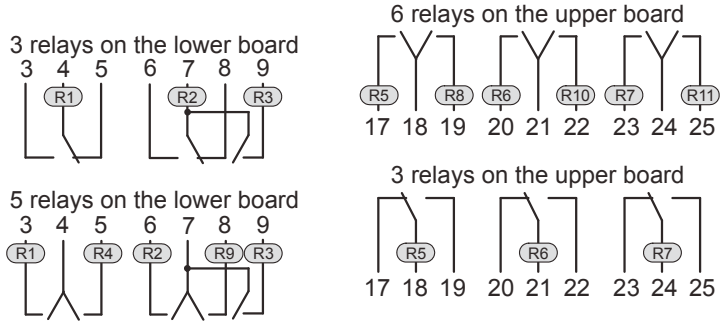
All the inputs can be configured for any type of sensors regardless in sequence (TC, RTD, current, voltage, potentiometer, etc).



This figure depicts the maximum connectable number of sensors. The sensors can be mixed by type and sequence. The software arranges the connectors (pins).

Figure 2

The Figure 3 shows the wiring of the relays. The software allocates the relays automatically. You can see here the allocations by type. When configuring the channels they occupy the relays in sequence. So it is necessary to order the proper relays for proper operation.


3 relays on the lower board
3 4 5 6 7 8 9
R1 R2 R3
5 relays on the lower board
3 4 5 6 7 8 9
R1 R4 R2 R9 R3
6 relays on the upper board
R5 R8 R6 R10 R7 R11
17 18 19 20 21 22 23 24 25
3 relays on the upper board
R5 R6 R7
17 18 19 20 21 22 23 24 25

With 1 relay heat, cool, etc	1	2	3	4				
With 2 relays HEAT-COOL Valve positioning	1	2	3	4	1	2	3	4
Relay	R1	R2	R3	R4	R5	R6	R7	R8

The control block occupies the relays by the configuration. The occupation occurs by the sequence of configuration. E.g. the control block1 is a HEAT-COOL type loop it occupies the R1 an R5 relays. The next control block can occupy the remains. Take account that the necessary relays must be ordered for the configuration.

Figure 3

1.4. Working principle

The KD9 controller may be freely configured by the menu. The hardware contains three boards. Take in account that the well-chosen set of boards could fulfil the requirements. You ought to study the MANUAL and the relay arrangement before ordering.

The lower board is the base of the controller, without this it does not work. The upper and middle boards can be chosen uniquely each and together. The ordering code can be seen in the table underneath.

											Universal controller with 10000 programsegment
KD9-	0	0	0	0	0	0	0	0	0	0	Base construction with 1 input + 2 relays,
lower panel	1										6 inputs
		1									3 relays (2 **Form C** + 1 **Form A**)
		2									5 relays (5 **Form A**)
		3									3 Ssd (OPC 12 V/10 mA)
		4									5 Ssd (OPC 12 V/10 mA)
upper panel			1								+3 relays (3 **Form C**)
			2								+6 relays (6 **Form A**)
			3								+3 Ssd (OPC 12 V/10 mA)
			4								+6 Ssd (OPC 12 V/10 mA)
				1							Loop Power Supply (isolated) (24 Vdc, 100 mA)
				3							0/4-20 mA 2. isolated analogue output (max. 600 ohm)
				4							0/1-5 V 2. isolated analogue output
				5							0/2-10 V 2. isolated analogue output
					3						0/4-20 mA 1. isolated analogue output (max. 600 ohm)
					4						0/1-5 V 1. isolated analogue output
					5						0/2-10 V 1. isolated analogue output
						1					RS232 not isolated
						2					RS485 isolated
						3					ETHERNET on MODBUS
middle panel							1				7 digital inputs
							2				7 digital inputs + adapter (not isolated)
							3				7 digital inputs + adapter (optically isolated)
								1			„Centronics" interface for ESC/P and CBM-920 type printers
									3		0/4-20 mA 3.,4. analogue outputs not isolated (max. 250 ohm)
									4		0/1-5 V 3.,4. analogue outputs not isolated
									5		0/2-10 V 3.,4. analogue outputs not isolated
										1	MMC inner interface
										2	MMC inner interface + panel mountable adapter

There are short specifications for software and hardware properties in the table. You can find properties by the codes. The software determines the relationship among the inputs and outputs. Therefore you can use these options which you have bought.

The tree-structured menu is user friendly. Some parameters are not displayed, depending on the protect mode setting, the option boards used or the enabled-disabled status. So, e.g. if you configure an RTD the parameters of cold-junction will not appear in the menu.

Instead of the complicated control structure it has a very simple configuration method which will be shown in the the **CONFIGURATION NAVIGATION DIAGRAM.**

You can configure the controller by keys placed on the front panel (bezel) or from a PC. The front panel display shows all the information you need in the configuration process.

The KD9 can accept the messages from outer instruments ad can send too by the digital inputs and outputs. You can set up control net through the communication ports among two or more controllers. On one RS485 line you can connect 31 slave and 1 master controller together. The PC can accept so many groups as the number its serial (RS232) ports.

The built in printer interface operates the matrix printer through parallel cable.

1.5. Settings using front panel keys (Handling)

The KD9 has 6 keys for configuring and using on the front panel (Figure 4). The keys have more functions. These will be shown there where they operate. Some second functions will be introduced below.

The **Second** key initializes the second function of a key. The red LED lights during its validity time. Need not holding down this key.

While the 2nd LED lights, you can turn in and out the controller by the **Exit** key. Please hold down the exit key for more than 5 s. After switching, the display will be flashing.

While the 2nd LED lights you can enter into the **Setpoint Programming** by the **Ente**r key.

While the 2nd LED lights you can switch on and off the **Manual mode** by the **Up** key, if it is enabled by **ConF./SYSt/mmi3[6], ConF./Cnt*/DECL[210]**variations.

While the 2nd LED lights you can skip a segment in the Setpoint Program by the **Right** key (ADVANCE).

While the 2nd LED lights you can stop and rerun the operation of the Setpoint Program by the **Down** key (HOLD/CONTINUE).

While holding down this key for a long time the auto-tune function initializes and **T** begins flashing.

When a change occurs the actual display begins flashing

changes the control block numbers (yellow 1,2,3,4), after holding down for 30s visualizes the PASS menu.

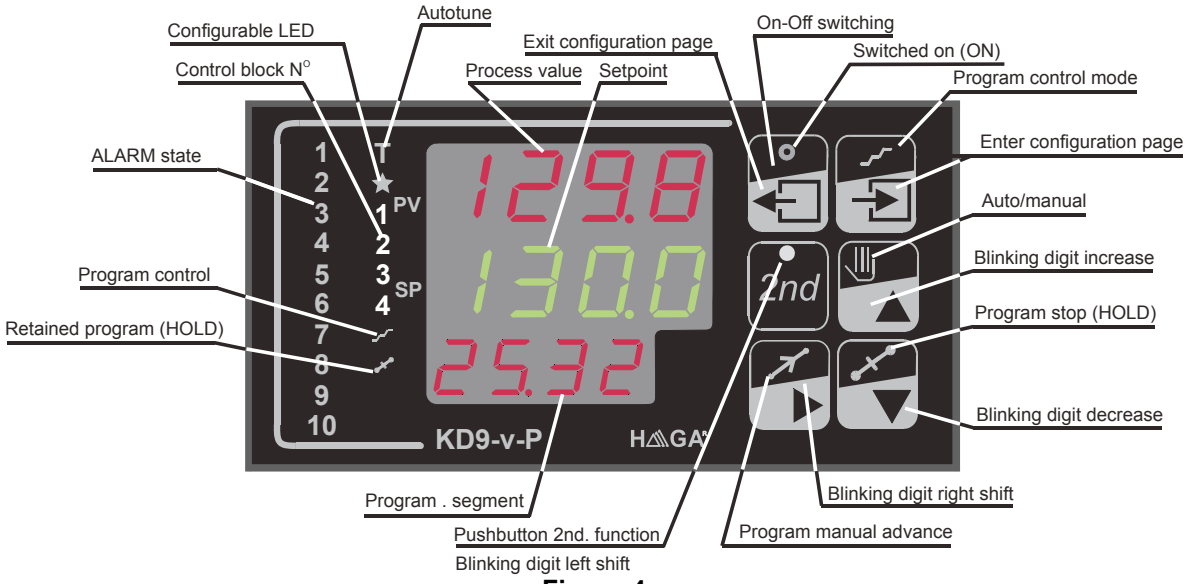


Figure 4

The digits and LED-s are always conform to operation. There are two operation modes:

Working Mode: ON state, **OFF** state and **Standby** state.

Setting Mode: **configuration** process, parameter **value** setting (you can configure in any state if it is enabled).

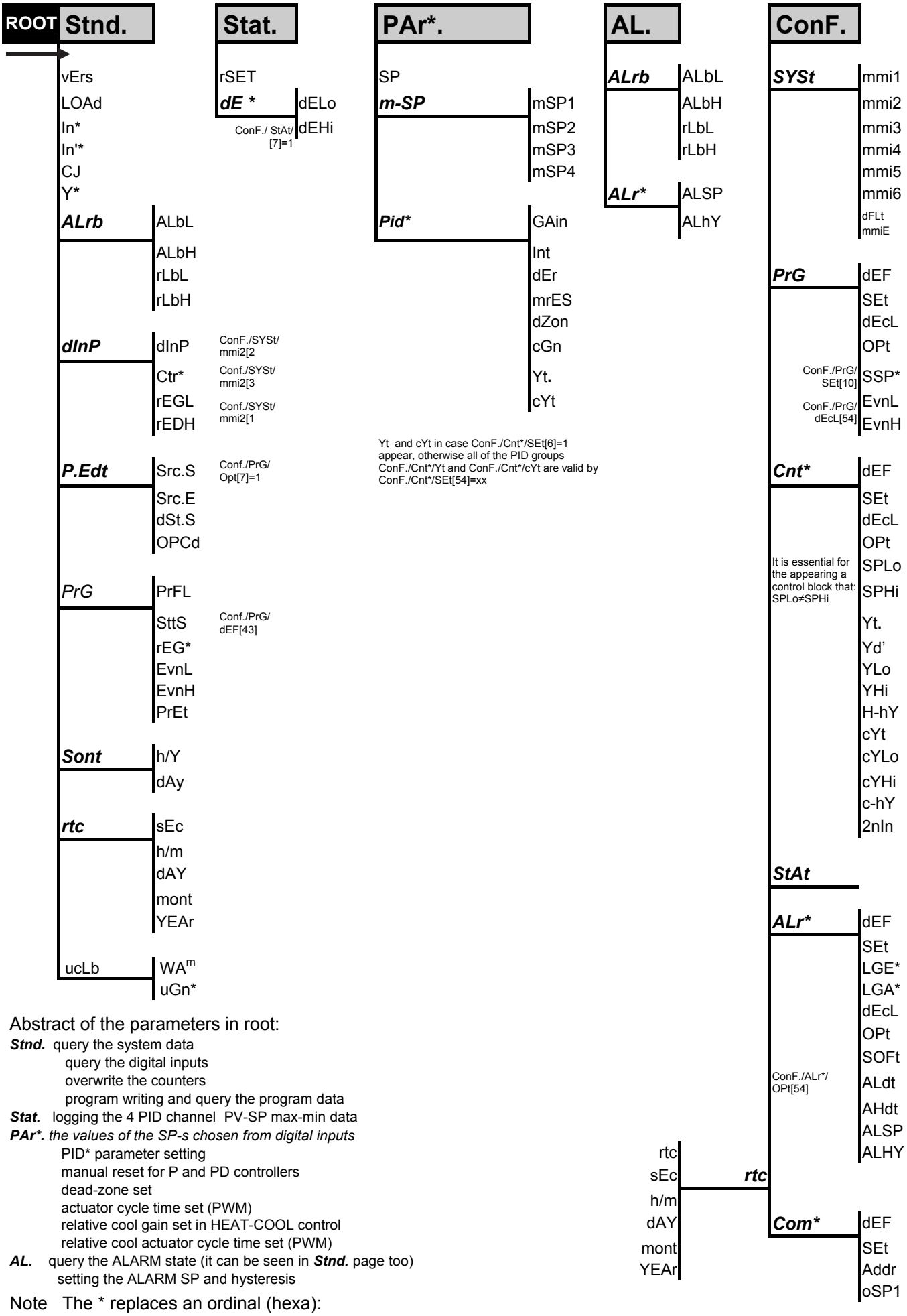
In Working Mode the displays show the data of the controlled system. There are the main numeric data and the mnemonics on the front panel. Where it is necessary more data are displayed alternatively. Every parameter is connected with its value: **What ↔ How many.**

You could configure the controller in **Setting Mode** while giving the parameters of control. You can see what are you configuring or give values. So there can be seen together all the data of the extraordinary complex system. **Where ↔ What ↔ How many.**

E.g. the **ALARM 11** latch properties can be configured with the bits at **Conf./AlrC/SEt**. When you reach this parameter 8 bits appear on the upper display and may be set them with the **Up**, **Down** and **Right** keys. In the middle display the **SEt** appears and in the lower display **Conf.** and **AlrC** alternate. You can configure the appropriate properties by the given table in this Manual setting the **EDS** (**E**lectronic **D**IP **S**witch). The **EDS** is an octernary switch group whit which you can select one property from 256 possibilities.

The parameters of a control channel appear in the three displays. There are configuration possibilities to change these parameters e.g. the green display can show one of the In* instead of the SP of the displayed channel (number 1..4 on left). So the displays may visualize three inputs at the same time (three In*). See in paragraph **Numeral display setting.**

1.6. CONFIGURATION NAVIGATION DIAGRAM /1 (hereafter "Navigation Diagram")



Abstract of the parameters in root:

Stnd. query the system data
query the digital inputs
overwrite the counters
program writing and query the program data

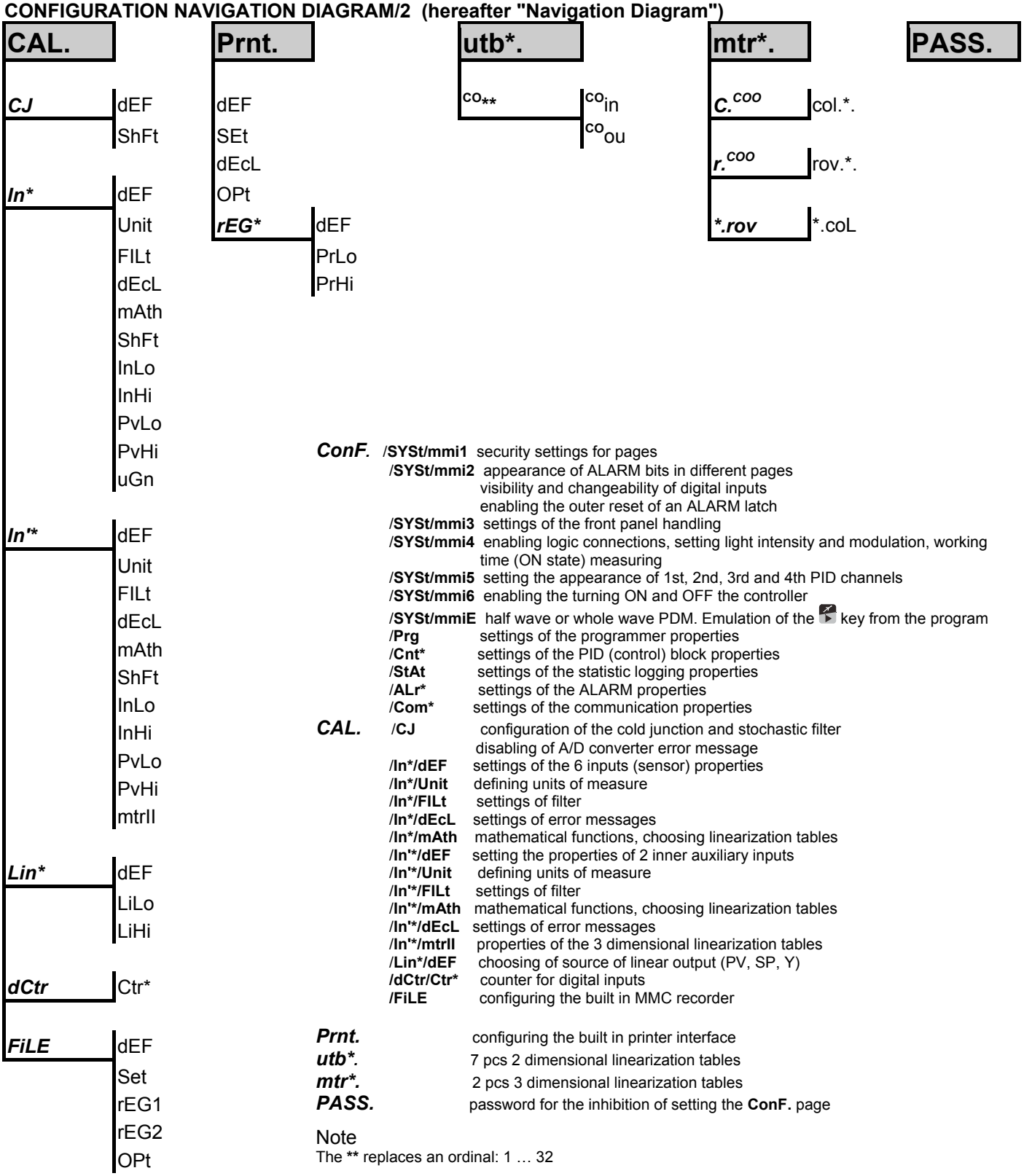
Stat. logging the 4 PID channel PV-SP max-min data

***PAr**.** *the values of the SP-s chosen from digital inputs*
PID* parameter setting
manual reset for P and PD controllers
dead-zone set
actuator cycle time set (PWM)
relative cool gain set in HEAT-COOL control
relative cool actuator cycle time set (PWM)

AL. query the ALARM state (it can be seen in ***Stnd.*** page too)
setting the ALARM SP and hysteresis

Note The * replaces an ordinal (hexa):
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F,G

CONFIGURATION NAVIGATION DIAGRAM/2 (hereafter "Navigation Diagram")



ConF. **/SYSt/mmi1** security settings for pages
/SYSt/mmi2 appearance of ALARM bits in different pages
visibility and changeability of digital inputs
enabling the outer reset of an ALARM latch
/SYSt/mmi3 settings of the front panel handling
/SYSt/mmi4 enabling logic connections, setting light intensity and modulation, working time (ON state) measuring
/SYSt/mmi5 setting the appearance of 1st, 2nd, 3rd and 4th PID channels
/SYSt/mmi6 enabling the turning ON and OFF the controller
/SYSt/mmiE half wave or whole wave PDM. Emulation of the key from the program
/Prg settings of the programmer properties
/Cnt* settings of the PID (control) block properties
/StAt settings of the statistic logging properties
/ALr* settings of the ALARM properties
/Com* settings of the communication properties

CAL. **/CJ** configuration of the cold junction and stochastic filter
disabling of A/D converter error message
/In*/dEF settings of the 6 inputs (sensor) properties
/In*/Unit defining units of measure
/In*/FILt settings of filter
/In*/dEcL settings of error messages
/In*/mAth mathematical functions, choosing linearization tables
/In'*/dEF setting the properties of 2 inner auxiliary inputs
/In'*/Unit defining units of measure
/In'*/FILt settings of filter
/In'*/mAth mathematical functions, choosing linearization tables
/In'*/dEcL settings of error messages
/In'*/mtrII properties of the 3 dimensional linearization tables
/Lin*/dEF choosing of source of linear output (PV, SP, Y)
/dCtr/Ctr* counter for digital inputs
/FiLE configuring the built in MMC recorder

Prnt. configuring the built in printer interface
utb*. 7 pcs 2 dimensional linearization tables
mtr*. 2 pcs 3 dimensional linearization tables
PASS. password for the inhibition of setting the **ConF.** page

Note
The ** replaces an ordinal: 1 … 32

Figure 5

1.7. Configuration

The microprocessor is the central unit of the controller (CPU). This microprocessor can work with the peripheral devices like a microcomputer for control systems. The inputs, outputs, display keys etc work as peripheral devices. As the microcomputer can do many functions, so the microprocessor based controller can control heat appliances, furnaces, drying chamber, packaging machine, play of light in fountain and many other system.

For working the whished mode, you must tell the controller how to do it. This knowledge you can write entirely in the configuration language. The pages are where to write and the mnemonics what to write in. The mnemonics, which appear in the menu, show the name of a parameter. One of the displays tells the value of the parameter. You can configure the controller by the keys on front panel or by the communication software.

So you purchased a universal control device from which you could "manufacture" a special device for your system. This process is called configuration.

The controller contains the ordered software. The default settings:

AL./ALr1/ ALSP = 100
AL./ALr1/ ALhY.= 5

PAr1./SP =. 100
PAr1./Pid/GAin =. 2,5
PAr1./Pid/Int = 240
PAr1./Pid/dEr = 50

Conf./Cnt1/dEF = [00000001]
Conf./Cnt1/SPHi = 300
Conf./Cnt1/Yt = 20
Conf./Cnt1/YHi = 100
Conf./ALr1/dEF = [00100000]

CAL./In 1/dEF = [00100000] (2 wire PT100)
CAL./In 1/FILt = [10000111]

All of the other configurational values ≡ 0

For easy configuring, you ought to study the Navigation Diagram and moving in the menu.

The Navigation Diagram

The menu items represent the properties of the controller. E.g., the item **CAL./CJ** is the place of the properties of the cold junction compensation. Here appear 8 bits, with which you can set how the cold junction will operate.

Due to the very logical program you could find easily the requested menu item. After finding the menu item in the Navigation Diagram, you can set everything by the tables containing the information you need. The Figure 5 shows the Navigation Diagram

The tree structure can be seen well in the diagram. The first row is the **ROOT** that comprises the main groups called pages. These mnemonics have a dot at its end (**Stnd. StAt. PAr*.** etc). You can find mnemonics for the parameters (menu items) in the second and lower level. The values of the parameters are at the lowest level. The "value" may be a number, an EDS or a mnemonic (it is clear by the definition of the menu item).

There are only 7 segments on the numeric display so the letters differ from the ordinary ones. The differences can be seen below

Letter	Display	Letter	Display	Letter	Display	Letter	Display
A		h H		o O		Y	
b		i I		P		c	
c C		J		r		o	
d		K		S		W	
E		L		t			
F		m		u			
G		n		v			

There are some equivalent elements in the controller e.g. the 4 PID loops, 16 ALARM-s etc. These elements are absolutely equal. The elements are marked with ordinals like **ALr 1, Alr 2, ALr 3, ALr 9, ALr A, … ALr G**. We use another signs at the linearisation tables: ^{oo} **1, …** ^{co}**32.** The ordinals are replaced by an * with a reference text which gives the range so: ^{co} *, where * = 1 … 32.

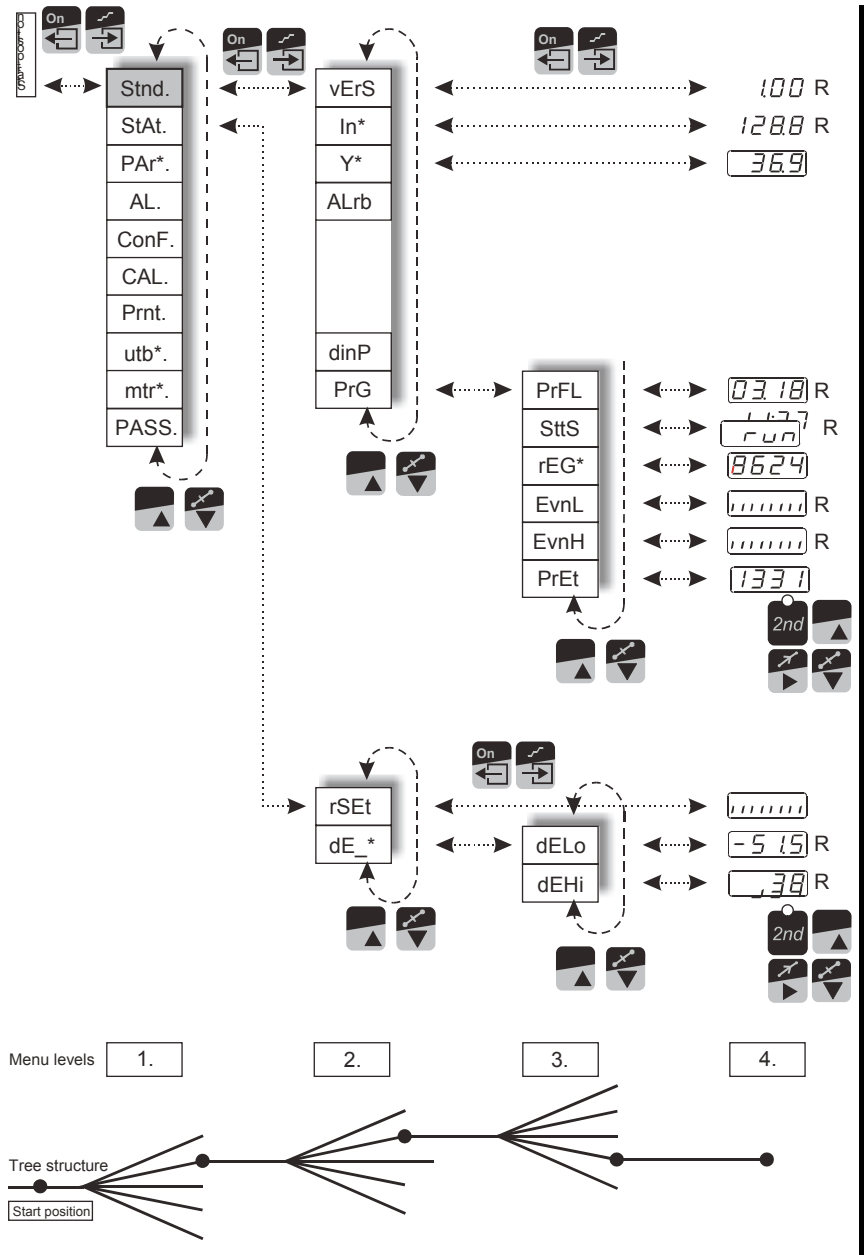
The mnemonics refers to the meaning of control properties if possible:

vErS version	**PrG** program	**PAr*** parameter	**SYSt** system
dEr derivate	**GAin** gain	**Int** integral	**mres** manual reset
mmi man machine interface	**dEF** definition	**SET** setting	**OPt** option
dEcL declaration	**rov** row	**coL** column	etc.

The tree structure can be seen as a book where the **Stnd.** is in the first page an the reminders are in the others. So we can speak about the **Stnd.** , **Stat.** etc pages as follows:

Stnd.	Standard page	**ConF.**	Configuration page	**utb*.**	Linearization table page
StAt.	Statistik page	**CAL.**	Calibration page	**mtr*.**	Matrix linearization table page
PAr*.	Parameter page	**Prnt.**	Printer page	**PASS**	Password page
AL.	ALARM value page	**File.**	File page (MMC card)		

1.8. Moving in menu



Special keys operation

The **Second** key initializes the second function of a key. The red LED lights during its validity time. Need not hold down this key.

While the 2nd LED lights, you can turn in and out the controller by the **Exit** key. Please hold down the exit key for more than 5 s. After switching the display, will be flashing.

While the 2nd LED lights you can enter into the **Setpoint Programming** by the **Ente**r key.

While the 2nd LED lights you can switch on and off the **Manual mode** by the **Up** key if it is enabled.

While the 2nd LED lights you can skip a segment in the Setpoint Program by the **Right** key (ADVANCE).

While the 2nd LED lights you can stop and rerun the operation of the Setpoint Program by the **Down** key (HOLD/CONTINUE).

While holding down this key for a long time the auto-tune function initializes and **T** begins flashing.

When a change occurs the actual display begins flashing

Figure 6

The structure and usage of menu (Figure 6)

You can browse among the levels with and in levels with . The numerical values and the EDS switches can be set with . During query and setting, the three numeric displays on front panel, show the actual parameter (menu item) and its value. The compound menu item is visualized by an alternating display. E.g. the programmer status appears on the upper display as alternating **run/11:27** and on the lower display alternating **Stnd/PrG.** On the middle display **SttS** is shown. Therefore, you can see all of the information you need.

If you are acquainted with browsing in menu, you are able to commission and use the controller.

The menu items could be ranked by function. There are **Read**-**Only** and **Write/Read** types. You can find those menu items, which are valid by configuration and can set if they are enabled to write. The **PASS** (password) is working by PC rules (detailed in particular section).

The Navigation Diagram shows only the place of the menu item in the tree. The detailed description can be found in configuration tables. The head of these tables contain the path, which refers to the Navigation Diagram. Where it is essential, we do examples to make the better understanding.

1.9. Menu item selection and giving parameter values

The following example illustrates the procedure of configuration. Let's configure the **ALARM 8** operating like this:

switch the relay (R8) after a 30 s period and latch if In3 reaches the value 350.

Menu items: find the **ConF./ALr*** (Figure 5). After setting **ConF./ALr8/SEt[0]** the **ConF./ALr8/dEF** table part 2 will be valid. Set the **ConF./ALr8/dEF = 00100011** value which will switch the **R8** state when the event occurs. The time relay can be configured by the **ConF./ALr8/OPt = 00110010.**

Value of parameters: You must give value to all configured parameters. Enter in the **AL.** page (Figure 5) and give value **AL./ALr8/ALSP = 350** it is the ALARM setpoint. In the same way set **ConF./ALr8/ALdt = 30** which defines the delay time.

This example demonstrates well the logic of configuration. After choosing a parameter, a value will be given for it. All of the parameters can be found in pages. The main stream: Find the parameters in Navigation Diagram and set them by the configuration tables.

1.10. Setting the numeric displays (the middle and the lower)

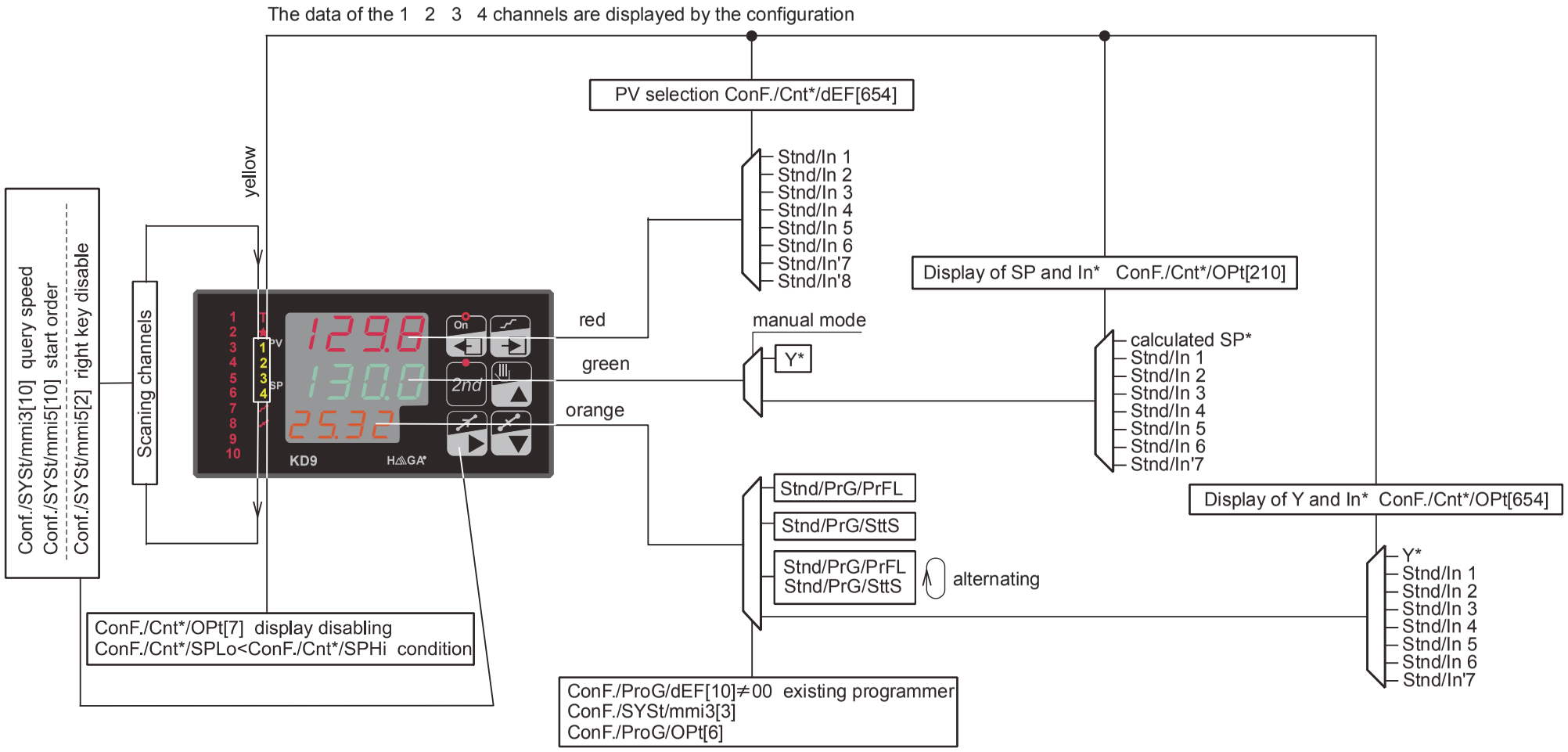
The controller can display data in three rows. One of the yellow numbers (1, 2, 3, 4) shows the validity of the four control (blocks) channels. Therefore, we may see 12 data by scanning or manual setting. The PV, SP and Y appear as defaults in the numeric display, but you can configure which data will appear in the middle and the lower display. The non-configured channels (blocks) do not appear.

The **RED** (upper) numeric display always shows the PV of the selected (1...4) channel with its unit if it is enabled and configured. Attention please! Those inputs (In*) containing units always appear everywhere with their units alternating in the display.

The **GREEN** (middle) display shows the SP of the control (blocks) channel in default. The Y is set here when the yellow number blinks in manual mode. We can show important values in the case of compound control task together as the different or equal SP-s. There are many other values, which can appear in this display such as inputs (In*), calculated SP, Y* in manual mode, etc. (Conf./Cnt*/OPt[210] for In*)

The **ORANGE** (lower) display shows the Y of the control (block) channel in default. The inputs seem here in proper setting to: Conf./Cnt*/OPt[654]. The data of the existing programmer can appear in the four orange displays which are set in ConF./SYSt/mmi5[43]. These settings overcome the original In* appearings. The display can show two data: the actual segment Stnd./PrG/PrFL or (and) the state of the running segment Stnd./PrG/SttS. We can control the display with ConF./SYSt/mmi3/[2] and ConF./PrG/OPt[6] switches.

Setting the numeric displays



1.11. Specifying linearization tables

Because of the nonlinearity of sensors output they must be linearized. There are 28 standard linearization tables built in the controller. Besides these tables, you can freely specify ten more tables. These are the user specified linearization tables. What are they for?

1. The valves are not linear because of their construction. When controlling mass flow you ought to linearize the characteristic. After specifying the table in the controller you could use in manual control mode. That means if you open the valve in manual mode to 50%, the flow rate will be 50% of total.
2. The liquid volume in an irregular shaped tank using a level sensor could be measured or controlled by the user specified linearization table.
3. ZIRCONIA - CARBON POTENTIAL CONTROL can be used after specifying a linearization table by the data given in its sensor manual.
4. It may happen that you can not change a sensor in a device, but you must change the controller. You can specify a linearization table for the sensor in the KD9.
5. The KD9 is able to control RH by psychrometric mode specifying the thermodynamic linearization table.

The following figures help the configuration.

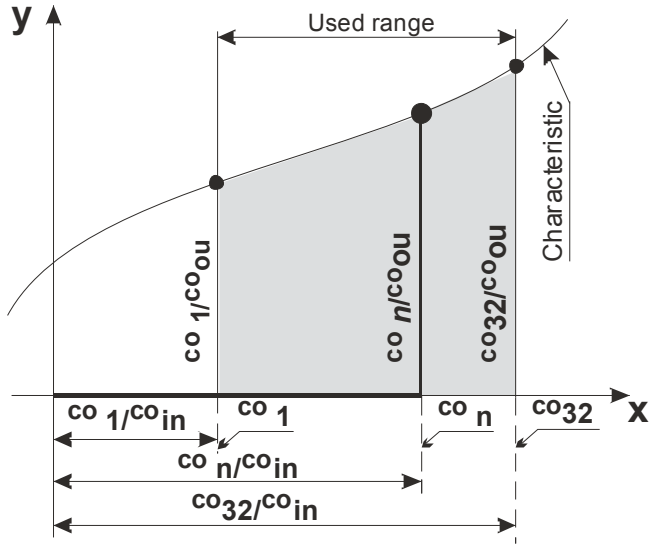


Figure 7

You can specify a 32 points defined characteristic curve in each table in the **utb*. (*=1 ... 7).** The program interpolates the internal points.
The x coordinates must be given in growing sequence with any closeness.
The values are the elements of a row matrix. The elements can be written or changed in any order.
The a(n) element is the y value of the x_n. The matrix contains max 32 elements.
In **utb*./co 1 ... utb*./co32** one of the 1 ... 32 numerals is the ordinal of the one point of the characteristic curve, called **n**. Two values are belong to it:

co n/co in x coordinate
co n/co ou y coordinate

You can install 7 characteristic tables in the KD9. The known characteristic curve of measuring instruments: metering orifice, pH gauge, volume of tanks (by height), infrared sensors, etc., are used for displaying their measured values in physical units.

You can determine 3 surfaces represented by 16x16 point each in **mtr***. (*=1, 2 and 3). The program interpolates the internal points. The values are the elements of an nxm type matrix. The columns and rows can be given in growing sequence with any closeness. The elements can be written or changed in any order.

Therefore the element a(nm) is in the n_{th} row and m_{th} column.

The meaning of the indexes:
n is the n_{th} point on the x axis
m is the m_{th} point on the y axis
nm-is the nm_{th} point on the xy plane
Where **n** and **m = 1 ... 9, A ... G**

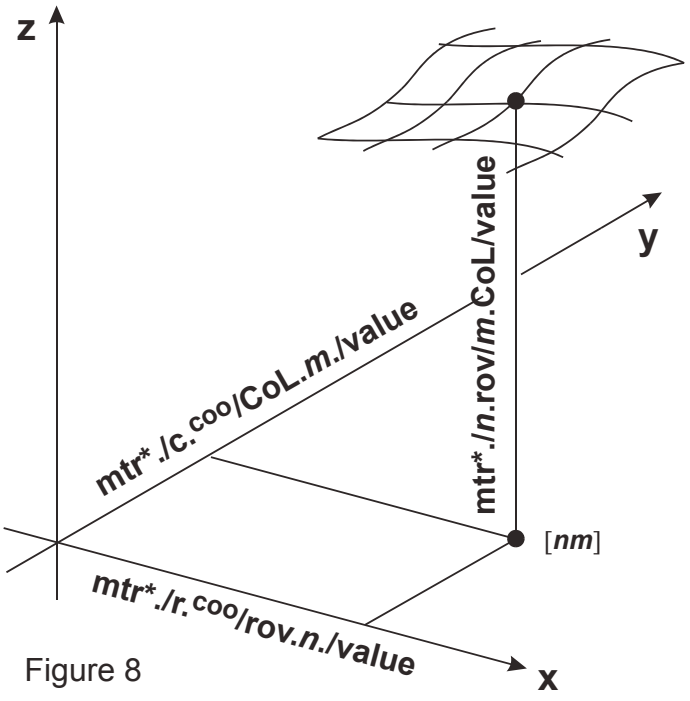


Figure 8

Example
For psychrometric RH measuring, we use two thermometers a dry and a wet one. In the RH tables in rows the dry bulb temperatures are written and in columns the differences. You can give the elements of the matrix as follows taking account of choosing the range of 16x16 points properly to the control task.

Write in rows (**r.^{coo}/rov.n.**) the dry bulb values by the table (e.g.: 24, 26, 28, 31, ...)
Write in columns (**c.^{coo}/Col.m.**) the dry-wet bulb difference values by the table (e.g.: 1,0; 1,5; 2,0; 3,0; 4,0; 6,0; ...)
Write in the nm_{th} element (**n.rov/m.CoL**) RH values by the table (e.g..: 92, 88, 85, 80, 75, 67, ...)

If you have chosen well the values, the KD9 will control very precisely.

The microprocessor reads data from the table ascending by the $\mathbf{n}_{th}$ and $\mathbf{m}_{th}$ ordinals. The program accepts the monotone ascending values: x(**n**)<x(**n+1**), y(**m**)<y(**m+1**), You can decrease the load of the processor by adding the necessary data only. So you ought to close the axes at the end of the used range. The close may be a pair of index values, where the second is smaller than the previous. x(**n**) > x(**n+1**) and y(**m**) > y(**m+1**). So it will be the last date in the table and limit of range. The values **n**=32 and **m**=16 close the axes automatically.

Configuration tips. (Connecting of linearization tables)

The **utb*.** (*=1 ... 7) and the **mtr*.** (*=1, 2 and 3) are the parts of the program of the controller. The controller uses these by the configuration after filling them. The number of elements is limited. It depends on the task if the number of elements is enough for the wanted accuracy. If the number of the elements is not enough, you can link the tables. So you can link 2 from the utb*. (*=1 ... 7) and 3 from the mtr*. (*=1, 2 and 3) by one event, which an ALARM can activate.

The **utb*.** (*=1 ... 7) tables can be connected together by changing over in CAL./In'7 and CAL./In'8. See: **Control block configuration** flowchart.

The **mtr*.** (*=1, 2 és 3) tables can be connected together by the next example:

In'7 auxiliary input

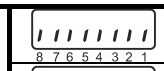
CAL./In'7/dEF[3210)=0010	let the first input the In2, its values will be on the X axis
CAL./In'7/mtrll[3210)=0011	let the second input the In3, its values will be on the Y axis
CAL./In'7/mtrll[4)=0	uses the **mtr1.** table, while **Stnd./ALrb/rLbH(3)**=0
CAL./In'7/mtrll[765)=011	assignes **Stnd./ALrb/rLbH(3)** for changing over, if it is =1, the **mtr3.** will be valid

In'8 auxiliary input, if the **mtr2.** table is needed

CAL./In'8/dEF[3210)=0010	let the first input the In2, its values will be on the X axis
CAL./In'8/mtrll[3210)=0011	let the second input the In3, its values will be on the Y axis
CAL./In'8/mtrll[4)=1	uses the **mtr2.** table, while **Stnd./ALrb/rLbH(4)**=0
Conf./SYSt/mmi5[7]=1	allowing the In2 with **ConF./Cnt*/2nIn** to **CAL./In'8**
Conf./Cnt*/dEF[654]=110	**In'7** will be the first input, X axis
Conf./Cnt*/2nIn[654]=111	**In'8** will be the second input, second axis
Conf./Cnt*/2nIn[3210]=1101	assignes Stnd**./ALrb/rLbH(4)** for changing over, if it is =1, the **mtr2**. will be valid

2. Base (parameter) tables

2.1. Stnd.

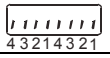
Query		Stnd.
vErS	Version number	
LOAd	Free processor resource in % (Enabled if: **ConF./SYSt/mmi2[3] = 1**)	
In* and **In'***	Input * value, where * = 1 ... 8	
CJ	The cold junction temperature (if enabled)	
Y*	Output * of PID block, where * = 1 ... 4 (can be changed in MANUAL mode)	
ALrb/ALbL	Result of the ALARM 1-8 function before logic operation; where EDS[0]=ALr1 EDS[7]=Alr8	
ALrb/ALbH	Result of the ALARM 9-G function before logic operation; where EDS[0]=ALr9 EDS[7]=AlrG	
ALrb/rLbL	Result of the ALARM 1-8 function after logic operation; where EDS[0]=ALr1 EDS[7]=Alr8	changes the relay state if it is not occupied for other pupose
ALrb/rLbH	Result of the ALARM 9-G function after logic operation; where EDS[0]=ALr9 EDS[7]=AlrG	
dInP/ dInP	State of digital input, where EDS[0]=Di1 … EDS[7]=Di8. Enabled if **Conf./SYSt/mmi2/[2] = 1** (only for testing)	
dInP/ Ctr*	Counting or time values of digital inputs. Changeable if **Conf./SYSt/mmi2/[3] = 1** (for testing)	
dInP/ rEGL	Register which can be set by PC for ALARM function source. rEGL[1…8]	The values of these registers are always readable and writable and usable as ALARM sources on MODBUS. They can be set here without PC too. 
dInP/rEGH	Register which can be set by PC for ALARM function source. rEGH[9…G]	
P.Edt /Src.S	The first segment of a part of program: Prog N°/Step N°	**The sort of operation:** (enabled by **ConF./PrG/OPt[7] = 1**)
P.Edt /Src.E	The last segment of a part of program: Prog N°/Step N°	**1** clear **5** clear without event code
P.Edt /dSt.S	The first place of the part of program when moving or copying: Prog N°/Step N°	**2** copy **6** copy without event code
P.Edt/OPCd	The sort of operation (**1**, **2**, **3**, **5**, **6, 7,**) The operation can be initialized by the key	**3** move **7** move without event code
PrG/PrFL	■■.□□ where the ■■ is the ordinal of SP-TIME program and the □□ is the ordinal of segment (program step)	
PrG/SttS	Status of programmer. E.g. Stnd./PrG/SttS = SOAK / 00:10 means that system is in soak since 10 minutes	
PrG/rEG*	Existing number in counter **rEG*,** where * **=** 1 ... 4, (can be overwrited if it is enabled in **ConF./ProG/dEF**[43])	
PrG/EvnL	State of the first 8 event	
PrG/EvnH	State of the second 8 event	
PrG/PrEt	The program start delay value. Adjustable in OFF state. Configurable to show the elapsed or remained time	
Sont/h/m	Operating period from the last default setting. **Conf./SYSt/dFLt = 89** resets to 0.	hour/minute
Sont/dAY		day
rtc/sec	Adjusting and reading the real time clock of the controller Enabling the adjustment in **ConF./rtc** **The default setting Conf./SYSt/dFLt = 89 deletes the exact time!** (returns to 02.01.2005) Only for Data-Acquisition-Cards and printer interface options	second
rtc/h/m		hour:minute
rtc/dAY		day
rtc/mont		month
rtc/YEAr		year
ucLb/WA^rn	**Warning!** The *special gain like setting* may cause harmful effects in the controlled system.	For calibrating the A/D converter.
ucLb/uGn*	*Special gain like setting* of an In*, where * = 1, ... 6. (It is the same as the **CAL./In*/uGn**)	

The **Standard Page** has many functions. The first is the **query** that serves for checking state of the system. Another important function the configuration of some menu items, like relocate program parts, changing the value of a register, etc.

The meaning of status values (PrG/SttS):

????	missing value	
SOAK	**:** ↔ "SOAK"	"SOAK" and soak time alternate
run	**:** ↔ "run"	"run" and time in this state alternate
run-	**:** ↔ "run-"	"run-" and time in this state alternate
PrEt	"PrEt"	in start delay state. The actual value is in **PrG/PrEt**
SLAv		SLAVE type controller
FLAG	"FLAG"	one of the FLAG -s menu item is valid in this segment

2.2. StAt.

StAt.		StAt
StAt./rSEt	0000**** clear data by logic =1, after its value = 0 ****0000 saving the last data freezes data logging by logic = 1. After zeroing data logging starts again	* is the ordinal of the control block 
StAt./dE*/dELo	The max PV-SP values of the PID block* since the last clear (**rSEt**), where * is the ordinal of the block	
StAt./dE*/dEHi	The min PV-SP values of the PID block* since the last clear (**rSEt**), where * is the ordinal of the block	

2.3. PAr.*

Parameter value		PAr*
SP	Adjusting setpoint of the control (PID) block (**ConF./Cnt***) or the query of configuration	
mSP/mSP1	Adjusting setpoints chosen from digital inputs (dInP[1] and dInP[0]) by **Stnd./dInP/dInP[10]** code combination if **Conf./Cnt*/SEt** [2] = 1	
mSP/mSP2		
mSP/mSP3		
mSP/mSP4		
Pid*/GAin	Gain of the PID* set in %. The proportional band in SP unit: p =100/GAin	where *=1... 9, A ... G
Pid*/Int	Integral (Reset) value of the PID*, set in s.	
Pid*/dEr	Derivative (Rate) value of the PID*, set in s.	
Pid*/mrEs	Manual reset value of the PID*, set in SP unit	
Pid*/dZon	Dead zone of the PID*, set in SP unit for HEAT-COOL and motorized valve control	
Pid*/cGn	Gain of the PID*, set in %. (For the cooler of HEAT-COOL control) ☼	
Pid*/Yt.	Actuator cycle time value of the PID* for PWM in s	
Pid*/cYt	Actuator cycle time value of the PID* for PWM in s (For cooler of HEAT-COOL control) ☼	

Max. 16 PID sets can be configured in the controller. The configuration place is **ConF./Cnt*/SEt[54]**, labelled with: **Pid1** … **PidG**

In case of **ConF./Cnt*/SEt[6] = 0** all of the PID set have the same **Yt.** and **cYt** values (the values of the PID1), which you can set at **Conf./Cnt*/ Yt.** and **Conf./Cnt*/ cYt**

In case of **ConF./Cnt*/SEt[6] = 1** all of the PID set may have different **Yt.** and **cYt** values, which you can set at **PAr*./Pid*/Yt.** and **PAr*./ Pid*/cYt**

☼ **c** is the multiplier constant of the cooler to heater power, *cooler gain* = **c** x *heater gain* *cooler cycle time* = **c** x *heater cycle time*

2.4. AL.

Query	Read only, the same as can be seen on **Stnd.** page	AL.
AL./ALrb/ALbL	Result of the ALARM 1-8 function before logic operation; where EDS[0]=ALr1 EDS[7]=Alr8	
AL./ALrb/ALbH	Result of the ALARM 9-G function before logic operation; where EDS[0]=ALr9 EDS[7]=AlrG	
AL./ALrb/rLbL	Result of the ALARM 1-8 function after logic operation; where EDS[0]=ALr1 EDS[7]=Alr8	changes the relay state if it is not occupied for other pupose
AL./ALrb/rLbH	Result of the ALARM 9-G function after logic operation; where EDS[0]=ALr9 EDS[7]=AlrG	

Value of parameter		AL.
AL./ALr*/ALSP	SP setting for **ALr***, where * = 1…G	
AL./ALr*/AHhY	Hysteresis setting for **ALr***, where * = 1…G	

The value of symmetric hysteresis is always a positive number. The value of lower asymmetric hysteresis is a negative, the upper a positive number.

3. ConF configuration tables

3.1. ConF./SYSt

7	6	5	4	3	2	1	0	**Con.F/SYSt/mmi1**	**ConF./SYSt**
							1	**Utb*.** page is visible	
						1		**mtr*.** page is visible	
					1			**Prn.** page is visible	
				1				**Stnd.** page is hidden if the PASS contains a password	
			1					**CAL.** page is hidden	
		1						**AL**. page is hidden	
	1							**PAr.** page is hidden	
1								**Stnd.** page is hidden	

7	6	5	4	3	2	1	0	**ConF./SYSt/mmi2**	**ConF./SYSt**
							1	The ALARM-bits are visible on the **Stnd.** page.	
						1		The digital input bits are visible on the **Stnd.** page. Bits of **rEGL** and **rEGF** are visible and adjustable in **Stnd.** page.	
					1			The digital input bits can be set by the keys on the **Stnd.** page, the rear contacts do not work (for testing the digital inputs)	
				1				**Stnd./dInP/Ctr*** can be set (if **CAL./dctr/Ctr*[7]** = 1)	
			1					Copies the value of **Stnd./ALrb/rLbH[7]** to the **Stnd./dInP/dInP[3]** place (**ALrG** relay state will be copied in the **dInP[3]** non existing digital input.)	
		1						The load of the microprocessor can be seen on the **Stnd.** page in %	
	1							The latches can be reset by the **dInP[0]** digital input	
1								Enables the outer ON-OFF switch by **dInP[2]** digital input	

7	6	5	4	3	2	1	0	**ConF./SYSt/mmi3**		**ConF./SYSt**
						0	0	Scan period for seeing a PID block	none	The PID block can be seen for the set period. The values scan after each other. They can be visualized by the key too.
						0	1		4s	
						1	0		8s	
						1	1		12s	
					1			Units can be assigned to inputs from **CAL./In*/Unit**		
				1				Y value can be seen on the lower display (in the place of Program /Segment Number) by **ConF./PrG/OPt[6]**		
			0					The EDS can be set by the right and by the up or down pressing either consequently		
			1					The EDS can be set by the right and by the up and by the down		
		0						Pressing down 2nd more than 12 s starts the autotune process (global)		
		1						Disables the property of the 2nd key to start the autotune process (global)		
	1							Disables the property of the key to toggle AUTO/MAN mode (global)		
1								Disables the property of the key to toggle ON/OF from the front panel		

7	6	5	4	3	2	1	0	**Conf/SYSt/mmi4**		**Conf./SYSt**
							1	Enables to set the enhanced ALARM logic functions properties (**Conf./ALr*/LGE1, LGE2, LGA1, LGA2**)		
						1		Total light on display		
			0	0	0			none	The light intensity alternation warns if an event occur. The light of display alternates from whole to half.	
			0	0	1			ALr6		
			0	1	0			ALr7		
			0	1	1			ALr8		
			1	0	0			ALr9		
			1	0	1			ALrA		
			1	1	0			ALrb		
			1	1	1			ALrC		
		0						Modulation frequency 0,5 Hz		
		1						Modulation frequency 3 Hz		
	0							**on** state	Measures the working time when it is in powered on state (the displays are lighting).	
	1							**on and OFF** state		
1								The measured working time value can be seen in **Stnd/Sont**.		

7	6	5	4	3	2	1	0	Conf/SYSt/mmi5	Conf./SYSt
						0	0	**1** yellow LED	The here configured channel data appear when power on the controller. If a channel does not exist, than the next one. The yellow LED shows the number of the visible channel. You can configure the appearing data of a channel in the **ConF./Cnt*/dEF[564]** and the **ConF./Cnt*/OPt[654 210]** places.
						0	1	**2** yellow LED	
						1	0	**3** yellow LED	
						1	1	**4** yellow LED	
					1			The channel scan toggle key is disabled	
			0	0				1 channel	You can send the SP program information in the orange (3rd) display with these bits and the proper yellow number will light (1 2 3 4). Other connected configuration places: **ConF./SYSt/mmi3[3]** and **ConF./PrG/OPt[76]** If the "programmer for 4 blocks" is running, you can configure different information for each of them **ConF./PrG/OPt[7654]**.
			0	1				2 channel	
			1	0				3 channel	
			1	1				4 channel	
		1						The gain like corection appears in **Stnd.** page and it is adjustable	
	1							Enables a gain like corection (**CAL./In*/uGn** and **CAL./In*/dECL[4]**)	
1								Enables another PV input in a channel by **ConF./Cnt*/2nIn**	

7	6	5	4	3	2	1	0	Conf/SYSt/mmi6	Conf./SYSt
on				**OFF**					
0	0	0	0					there is not a turn **on**	This configuration turns the controller **on** by an ALARM function event. (See the ALARM function diagram) Example: in case of **Conf/SYSt/mmi6[7654]=0101**, if **Stnd./ALrb/rLbL[4]=1** the controller turns **on**. So when you want to turn **on** the controller, choose the bits of **Stnd/ALrb/rLbL [4]** and when the **Stnd./ALrb/rLbL[4]=1** ALARM condition will be fulfilled the controller get to **on** state Note:- The three states of the controller: 1. No-voltage, the displays do not light. 2. **on**, the "On key" red LED lights. 3. **OFF**, the "On key" red LED does not light. The **Stnd./ALrb/rLbL** and the **Stnd./ALrb/rLbH** are the binary functions of ALARM-s. The controller always counts the function value independently of an existing physical input. When a channel (control block) allocates automatically a relay the controller due its selectivity can use their **Stnd./ALrb/rLbL** and **Stnd./ALrb/rLbH** functions.
0	0	0	1					**Stnd./ALrb/rLbL [0]**	
0	0	1	0					**Stnd./ALrb/rLbL [1]**	
0	0	1	1					**Stnd./ALrb/rLbL [2]**	
0	1	0	0					**Stnd./ALrb/rLbL [3]**	
0	1	0	1					**Stnd./ALrb/rLbL [4]**	
0	1	1	0					**Stnd./ALrb/rLbL [5]**	
0	1	1	1					**Stnd./ALrb/rLbL [6]**	
1	0	0	0					**Stnd./ALrb/rLbL [7]**	
1	0	0	1					**Stnd./ALrb/rLbH[0]**	
1	0	1	0					**Stnd/.ALrb/rLbH[1]**	
1	0	1	1					**Stnd./ALrb/rLbH[2]**	
1	1	0	0					**Stnd./ALrb/rLbH[3]**	
1	1	0	1					**Stnd./ALrb/rLbH[4]**	
1	1	1	0					**Stnd./ALrb/rLbH[5]**	
1	1	1	1					**Stnd./ALrb/rLbH[6]**	
				0	0	0	0	there is not a turn **OFF**	This configuration turns the controller **OFF** by an ALARM function event. (See the ALARM function diagram) Example: in case of **Conf/SYSt/mmi6[3210] = 1011**, if **Stnd./ALrb/rLbL[2]=1** the controller turns **OFF**. So when you want to turn **OFF** the controller, choose the bits of **Stnd/ALrb/rLbL [2]** and when the **Stnd./ALrb/rLbL[2]=1** ALARM condition will be fulfilled the controller get to **OFF** state Note:- The three states of the controller: 1. No-voltage, the displays do not light. 2. **on**, the "On key" red LED lights. 3. **OFF**, the "On key" red LED does not light. The **Stnd./ALrb/rLbL** and the **Stnd./ALrb/rLbH** are the binary functions of ALARM-s. The controller always counts the function value independently of an existing physical input. When a channel (control block) allocates automatically a relay the controller due its selectivity can use their **Stnd./ALrb/rLbL** and **Stnd./ALrb/rLbH** functions.
				0	0	0	1	**Stnd./ALrb/rLbL [0]**	
				0	0	1	0	**Stnd./ALrb/rLbL [1]**	
				0	0	1	1	**Stnd./ALrb/rLbL [2]**	
				0	1	0	0	**Stnd./ALrb/rLbL [3]**	
				0	1	0	1	**Stnd./ALrb/rLbL [4]**	
				0	1	1	0	**Stnd./ALrb/rLbL [5]**	
				0	1	1	1	**Stnd./ALrb/rLbL [6]**	
				1	0	0	0	**Stnd./ALrb/rLbL [7]**	
				1	0	0	1	**Stnd./ALrb/rLbH[0]**	
				1	0	1	0	**Stnd./ALrb/rLbH[1]**	
				1	0	1	1	**Stnd./ALrb/rLbH[2]**	
				1	1	0	0	**Stnd./ALrb/rLbH[3]**	
				1	1	0	1	**Stnd./ALrb/rLbH[4]**	
				1	1	1	0	**Stnd./ALrb/rLbH[5]**	
				1	1	1	1	**Stnd/ALrb/rLbH[6]**	

Parameter value	*Default settings and special clearings*	ConF./SYSt/dFLt
32	Deletes the linearisation tables: **utb*.** and **mtr*.**	
57	Changes the parameters to default set in the controller, without the linearisation tables	
89	Changes all of the parameters to default set. **The real time clock goes to base state.** Adjust the real time.	
102	Resets the ALARM latches	
128	Clears the error messages if the error does not exists	
173	Clears the saved programs from programmer memory	
196	Clears the saved programs where PrFL≥50 and the SP2, SP3 and SP4 values of the same area saved 4 profile programs	
212	Clears the counters of the digital inputs	

7	6	5	4	3	2	1	0	Con.F/SYSt/mmiE	ConF./SYSt
					0	0	0	There is not a profile connected program step enforcement (function of key).	
					x	x	x	Program step enforcement. Will be valid at the leading-edge of the assigned relay, where xxx=[001…111] = [Alr2…Alr8]. This ALARM must be allocated to the proper program step.	
				0				**P**ulse **D**ensity **M**odulation generates half waves, if Yt=0. One period of the 50 Hz mains contains 100 half wave.	
				1				**P**ulse **D**ensity **M**odulation generates whole waves, if Yt=0. One period of the 50 Hz mains contains 50 half wave.	

The Default Parameter set

AL./ALr1/ ALSP = 100
AL./ALr1/ ALhY.= 5

PAr1./SP =. 100
PAr1./Pid/GAin = 2,5
PAr1./Pid/Int = 240
PAr1./Pid/dEr = 50

Conf./Cnt1/dEF = [00000001]
Conf./Cnt1/SPHi = 300
Conf./Cnt1/Yt = 20
Conf./Cnt1/YHi = 100
Conf./ALr1/dEF = [00100000]

CAL./In 1/dEF = [00100000] (2-wire PT100)
CAL./In 1/FILt = [10000111]

ALL the others ≡ 0

3.2. SP Program configuration

ConF./PrG

7	6	5	4	3	2	1	0	**ConF./PrG/dEF**	**ConF./PrG**
						0	0	Programmer does not exists	
						0	1	Sequencer operation (without SP)	
						1	0	Programmer generates SP only for the 1. PID block. The others operate with their own SP-s.	
						1	1	Programmer for 4 blocks works with the same time-base, with 4 independent SP (SP1, SP2, SP3, SP4). In condition 0 ≤ PrFL < 50 only.	
					0			Every command is synchronized (1 command / 1s)	
					1			**SnoP** and **FrEE** commands are synchronized only. The program executes max 5 commands in 1 s from others	
				1				The counter values can be changed on the front panel (test and debug) **Stnd./PrG/rEG***	
			1					Counters are visible **Stnd./PrG/rEG***	
		1						Enabling the weekend timer (start delay) **Stnd./PrG/PrEt**	
	1							Enabling **HOLD** function from (Stnd./dInP/dInP[4] digital input **Di5 EDS[4]**. If **0** operates by "down" key (see Figure 4)	
#								Reserved for development	

7	6	5	4	3	2	1	0	**ConF./PrG/SEt**	**ConF./PrG**
						0	0	After turn on (**On**) → **SP*** = **PV***, the program starts from the actual **PV*** value. After turn off (**OFF**) → the display shows the last **SP*** and the last event code set is valid	
						0	1	After turn on.(**On**) → **SP*** = **SSP*** the program starts from the value saved in **SSP*** parameter After turn off (**OFF**) → the display shows the last **SP*** and the last event code set is valid	
						1	0	After turn on (**On**) → **SP*** = **PV***, the program starts from the actual **PV*** value. After turn off (**OFF**) → **SP*** = **PV***, changes continually the **SP*** value by the actual **PV*** and the **EvnL EvnH** event code sets are valid	
						1	1	After turn on (**On**) → **SP*** = **SSP*** the program starts from the value saved in **SSP*** parameter After turn off (**OFF**) →. the display shows the **SSP*** and the **EvnL** and **EvnH** event code sets are valid	
				0	0			****.00** the program starts from here (from where the program was last written or edited. From profile number)	The last set profile/segment is always in the orange display if **Conf./ProG/OPt/[5] = 1**
				0	1			****.**** the program starts from here (from where the program was last written or edited. From profile/segment number)	
				1	0			**00.00** the program starts from here (From profile/segment number = 00.00)	
				1	1			****.00** the program starts from digital inputs by the variation of Di(8, 7, 6) EDS[765]contacts, wherel ** = 00, 01, … 07 (see in section ***Digital Inputs***)	
			1					Turns to **OFF** state after power failure	
		1						There is not an **AUTOWAIT** function, the actual **SP** does not wait for the **PV**-t (do not wait until catching PV up)	
	1							NEXT key disabled (Figure 4)	
1								HOLD key disabled (Figure 4)	

7	6	5	4	3	2	1	0	**ConF./PrG/dEcL**	**ConF./PrG**
							1	**FLAG** type parameters are read only (but operate)	
						1		**SOAK** type parameters are read only (but operate)	
					1			**rAmP** type parameters are read only (but operate)	
				1				**timE** type parameters are read only (but operate)	
			1					The event code does not appear when writing the program	
		0						Every profile has an own event code set	
		1						Every profile uses the event code set of the profile 00	
0	0							All the SP program parameters always can be set	
0	1							SP program parameters can be set only in **OFF** state	
1	0							SP program parameters are read only	
1	1							You can set only the profile number, for starting a program	

7	6	5	4	3	2	1	0	**ConF./PrG/OPt**		**CnF./PrG**
								timE or **SOAK**	**rAmP**	The-timbase can be changed while running the progam with the **tb''**, **tb'**, **tb-h FLAG** commands
						0	0	Time-base minute **:** second	unit/minute	
						0	1	Time-base hour**:** minute	unit/hour	
						1	0	Time-base day **:** hour	unit/day	
					0			When all the 1 position EDS bits of an **IF FLAG** equal to all 1 position EDS bits of **Stnd./dInP/dInP**, the **IF FLAG** will be active (logic AND)		The rule is valid for FLAGs underneath: **IFi IFAL IFAH IFrL IFrH IFtn**
					1			When either of the 1 position EDS bits of an **IF FLAG** equals to one appropriate 1 position EDS bits of **Stnd./dInP/dInP**, the **IF FLAG** will be active (logic OR)		
				1				Programmer is in SHADOW working mode (for 1 profiled program mode only, there is not AUTOWAIT or HOLD)		
			1					After turn OFF in not standby state all of the event codes will be zero. **Stnd./PrG/Evn* = 0**		
		1						Rewrites the orange (lower) display to the last set profile/segment value. This is the same as **Stnd./PrG/PrFL**.		
1								Enables the editing o te program (**Stnd./P..Edt/OPCd**)		
								visible values in the orange (lower) display ↓		
	0							**Conf./SYSt/mmi3[3]=0**	profile/segment is displayed in **.** form	**Stnd./PrG/PrFL**
								Conf./SYSt/mmi3[3]=1	output signal (Y) is displayed in ***.* form	**Stnd./Y***
	1							**Conf./SYSt/mmi3[3]=0**	profile/segment in **.** form segment in text form and the past time in this here in **** form are displayed alternate	**Stnd./PrG/PrFL** **Stnd./PrG/SttS**
								Conf./SYSt/mmi3[3]=1	segment in text form and the past time in this here in **** form are displayed alternate	**Stnd./PrG/SttS**
1								Enabling program procedures (clear, copy, move) in page **Stnd./P.Edt/****		

The last written profile/segment will always appear in the orange (lower) display if **ConF./PrG/OPt[5]=1.**

Parameter value		**ConF./PrG**
Conf./PrG/SSP*	Program start from a value stored in this memory. If **Conf./PrG/SEt[0] = 1** can be seen and set	
Conf./PrG/EvnL	Setting the first 8 event code. If **Conf./PrG/SEt[1] = 1** can be seen and set	These setting are valid in **OFF** state of the controller
Conf./PrG/EvnH	Setting the second 8 event code. **Conf./PrG/SEt[1] = 1** can be seen and set	

3.3. Channel configuration

7	6	5	4	3	2	1	0	**ConF./Cnt*/dEF** where * = 1, 2, 3 and 4	**ConF./Cnt***
						0	0	There are not relay or SSd outputs	
						0	1	Control with **one** relay or SSd	
						1	0	HEAT_COOL control. In case of **ConF./Cnt*/dEF[210] = 110** without relay output, instead of the two occupied relays you can send the control output to a linear output in form Y=-100→100 % in **CAL./Lin*/dEF[43210]**=011xx	
						1	1	Motorized valve positioner	
					0			Heating control (inverse control)	
					1			Cooling control (direct control)	
				1				Enabling the **max value Y** for actuator when **Stnd./ALrb/ALbL**[*]= 1 (where * = 0 ... 3 for control blocks 1 ... 4) The actual Y value can not exceed the value stored in **Conf./Cnt*/Yd'**. Or cross-connected motor driver for valve positioner when using it. (xxxxxx11). Closes the valve at mains failure.	
	0	0	0					Input of the Control Block* **In 1**	where * = 1, 2, 3 and 4 the here chosen input (PV) will be the input at *control block.
	0	0	1					Input of the Control Block* **In 2**	
	0	1	0					Input of the Control Block* **In 3**	
	0	1	1					Input of the Control Block* **In 4**	
	1	0	0					Input of the Control Block* **In 5**	
	1	0	1					Input of the Control Block* **In 6**	
	1	1	0					Input of the Control Block* **In'7**	
	1	1	1					Input of the Control Block* **In'8**	
1								The Control Block* is always **on** state, where * = 1, 2, 3 and 4	

7	6	5	4	3	2	1	0	**ConF./Cnt*/SEt** where * = 1, 2, 3 and 4	**ConF./Cnt***
						0	0	SP source selection **SP+0** normal, without adding	
						0	1	SP source selection **SP+In 6** for using remote SP, cascade, ΔT cascade (outer), etc.	
						1	0	SP source selection **SP+In'7** for using mathematical relations, cascade, ΔT cascade (inner). etc	
						1	1	SP source selection **SP+In'8** for using mathematical relations, cascade, ΔT cascade (inner). etc	
					1			SP chosen from digital inputs Di2- Di1	
				1				Adding the computed value of **SP1** to the **SP*** of the control block* (**PAr*/SP**), where * = 2, 3 and 4 (relative offset from the 1. control block)	
		0	0					There is only 1 PID set	
		0	1					There are 4 PID sets with outer selection (PID* to the **m-SP/SP***, where * = 1, 2, 3 and 4)	
		1	0					There are 16 PID sets with selection by the actual SP (spaced equally among the configured **SPLo** and **SPHi**)	
		1	1					There are 16 PID sets with selection by the **EvnH[7654**] event codes (in program, by control block, by profile number/by segment number) where* = 0000=1…1111=G ¤	
	1							Every PID set has an own **Yt**. and **cYt**	
1								PID sets are invisible	

¤¤ **Note:** The bits of the **EvnH[7654**] could actuate **ALr***-s also, if they are defined at **ConF**./ **ALr*/dEF[10011xxx]** with the E8, E7, E6, E5 bits

7	6	5	4	3	2	1	0	**ConF./Cnt*/dEcL** where * = 1, 2, 3 and 4	**ConF./Cnt***
							1	Automatically switches to manual mode when an input failure occurs	
						1		Switches to manual mode by the digital inputs **Stnd./dInP/dInP[7654]** from the rear connectors (**Di8** ...**Di5**)	
					1			Enabling manual mode. Does not switch off the control block* when an input error is detected. **Be careful! Dangerous operation!**	
				1				Enables the linearly set of the output signal between **ConF./Cnt*/ YLo** and **YHi** values. It works only in relay and SSd mode, but does not work in manual mode. ¤¤¤	
			1					Inverts the **ConF./Cnt*/dEF[2]** bit, if the value of **Stnd./ALrb/ALbL**[*] = 1, where * = 0 … 3, in the 1 ... 4 control blocks so the heating will be cooling and v.v.	
		1						The autotune can be initialized from the front panel	
	1							Disables the adjusting of **SP** from the front panel and from **PAr*/SP**	
1								**PAr*** is invisible	

¤¤¤ **Note:** You can modify and scale the output signal. The next correlation exists between the PID algorithm computed parameters and the valid output signal:

Y=0% the relay or SSD switches by the duty factor written in **YLo**

Y=100% the relay or SSD switches by the duty factor written in **YHi,** and the inequality **YLo** < **YHi** must be fulfilled.

The duty factor is linear between these values.

The scaleable output signal is a very important property. There are controlled systems in which the built in power is too big. In such cases the steady state duty factor is very low and because of this the control will be wrong. Using this option, lower the **YHi** value, the duty factor will grow at steady state and the control will be better. It seems like that the built in power were reduced.

After every using of this option the system must be tuned again.

VERY INPORTANT! Do not set too high value for **YLo** because this is the minimum actuating and this may be dangerous for the system.

7	6	5	4	3	2	1	0	**ConF./Cnt*/OPt** where * = 1, 2, 3 és 4	**ConF./Cnt***
					0	0	0	Default is valid (SP)	**In(x)** input appears in the place of SP of the control loop (green display) (where **x = 1…7**), the SP exists in the **PAr*** page in the future too. The SP adjustment must be disabled by **Conf/Cnt*/dEcL [6]=1**.
					0	0	1	**In1**	
					0	1	0	**In2**	
					0	1	1	**In3**	
					1	0	0	**In4**	
					1	0	1	**In5**	
					1	1	0	**In6**	
					1	1	1	**In7**	
				0				A type derivative will working (derivative time will be calculated on "error", SP-PV)	
				1				B type derivative will working (derivative time will be calculated on "PV", (ΔPV)	
	0	0	0					Default is valid (Y)	**In(x)** input appears in the place of Y of the control loop (orange display) (where **x = 1…7**), the Y exists in the **Stnd/Y*** in the future too. In case of working an SP programmer, its state may override the **In(x)** here (in the orange display) by the value of **Stnd/SttS**. See: **Conf./SYSt/mmi3[3]**
	0	0	1					**In1**	
	0	1	0					**In2**	
	0	1	1					**In3**	
	1	0	0					**In4**	
	1	0	1					**In5**	
	1	1	0					**In6**	
	1	1	1					**In7**	
1								The data of **Cnt*** do not appear in the displays (**2**, **3** and **4** yellow numbers and the other data of this channel). If the entire 4 channel are disabled, the data of the **1** channel will appear here.	

Parameter value			**ConF./Cnt***
SPLo	The control block does not accept smaller SP than this		
SPHi	The control block does not accept bigger SP than this		
Yt.	The cycle time of the output signal for PWM (Pulse Width Modulation). The whole rotating time of valve positioner from opened to closed. (Set range:1-255 s) If Yt. = 0 the actuator will work with minimal ripple of power, for SSd usage only.		
Yd'	Under in **Yd'** stored value the motor does not move the valve. (proposed value 1-5). The maximal output signal (Y) except the motorized valve control		
YLo	The lower limit of the output signal, under value stored here Y=0.0 %	The computed value of **Y** is on the display. In case of an error YLo ≥ YHi automatically resets to: YLo=0, YHi=100	
YHi	The upper limit of the output signal, above value stored here Y=100.0 %		
H hY	The hysteresis of HEAT in ON-OFF control when HEAT-COOL control is configured. It is symmetric to the HEAT side of the dead zone		
cYt	The cycle time of the cooler output signal for PWM (Pulse Width Modulation), when HEAT-COOL control is configured.		
cYLo	The lower limit of the heater output signal, under value stored here Y=0.0 %	Operates same as: **YLo, YHi**	
cYHi	The upper limit of the cooler output signal, above value stored here Y=100.0 %		
c hY	The hysteresis of COOL in ON-OFF control when HEAT-COOL control is configured. It is symmetric to the COOL side of the dead zone		

7	6	5	4	3	2	1	0	**ConF./Cnt*/2nIn** where * = 1, 2, 3 és 4	**ConF./Cnt***
Enable in **ConF./SYSt/mmi5[7]=1.** The configured input can be changed to a 2nd input by an ALARM									
-	-	-	-	0	0	0	0	**there is not a 2nd PV**	If **Stnd/ALrb/rLbL [*]=1**, then the here set **In*** will be the input of the control block by the **ConF./Cnt*/2nIn[654]** bits. It does not change if this **In*** is not valid.
				0	0	0	1	**Stnd./ALrb/rLbL [0]**	
				0	0	1	0	**Stnd./ALrb/rLbL [1]**	
				0	0	1	1	**Stnd./ALrb/rLbL [2]**	
				0	1	0	0	**Stnd./ALrb/rLbL [3]**	
				0	1	0	1	**Stnd./ALrb/rLbL [4]**	
				0	1	1	0	**Stnd./ALrb/rLbL [5]**	
				0	1	1	1	**Stnd./ALrb/rLbL [6]**	
				1	0	0	0	**Stnd./ALrb/rLbL [7]**	
				1	0	0	1	**Stnd./ALrb/rLbH[0]**	
				1	0	1	0	**Stnd./ALrb/rLbH[1]**	
				1	0	1	1	**Stnd./ALrb/rLbH[2]**	
				1	1	0	0	**Stnd./ALrb/rLbH[3]**	
				1	1	0	1	**Stnd./ALrb/rLbH[4]**	
				1	1	1	0	**Stnd./ALrb/rLbH[5]**	
				1	1	1	1	**Stnd./ALrb/rLbH[6]**	
	x	x	x					the same as **ConF./Cnt*/dEF[654]**	
1								**Y** is alwais **0** if **Stnd/ALrb*=1 (Similar to Conf./Cnt*/dEF[3])**	

StAt

7	6	5	4	3	2	1	0	**ConF./StAt**	**ConF./StAt**
							1	Min-max of 1.control block	Min-max operates in ON state only
						1		Min-max of 2.control block	
					1			Min-max of 3.control block	
				1				Min-max of 4.control block	
			1					Enabling to freeze Min-max (see **StAt./rSEt**)	
		1						Disabling the reset function from menu	
	1							The statistical function invisible in the menu	
1								The statistical function enabled	

3.4. ALARM configuration

7	6	5	4	3	2	1	0	**ConF./ALr*/dEF** where * = 1, 2, 3, … 9, … A, b, C, d, E, F, G	**ConF./ALr***
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	This ALARM does not exist (all functions are disabled)	
The Process, Process-ratio, Deviation, Band type ALARM function **ConF./ALr*/SEt[6]** bit sets the visibility of hysteresis									
0								Symmetrical hysteresis	The SP of ALARM assigned an analogue input can be set in **AL./ALr*/ALSP,** the hysteresis in **AL./ALr*/ALhY** and the query about them by the bits of **ConF./ALr*/SEt**[76].
1								Asymmetrical hysteresis	
	0	1	0	0	x			Process **In**[x] (where x: 000→1, 001→2,…111→8 is the input number)	
	0	1	0	1	0	x		Process **SP**[x] (where x: 00→1, 01→2, 10→3, 11→4 is the SP number)	
	0	1	0	1	1	x		Process **Y**[x] (where x: x: 00→1, 01→2, 10→3, 11→4 is the Y number)	
	0	1	1	0	x			Process-ratio **PV**[x] (where x: 000→1, 001→2,…111→8 is the input number)	
	0	1	1	1	0	x		Process-ratio **SP**[x] (where x: 00→1, 01→2, 10→3, 11→4 is the SP number)	
	0	1	1	1	1	x		Process-ratio **Y**[x] (where x: 00→1, 01→2, 10→3, 11→4 is the Y number)	
	1	0	y		x			Deviation **SP**[y]-**In**[x] > **AL./ALr*/ALSP**	
	1	1	y		x			Band |**SP**[y]-**In**[x]| > **AL./ALr*/ALSP**	
2nd part of the **ConF./ALr*/dEF** table. (The **2nd** part is valid when **ConF./ALr*/SEt[6] = 0**)									
0	0	0	0	0	0	0	1	The ALARM always is **0**. (It can be used for reset latches and for other task when it is called in program or digital)	
0	0	0	0	0	0	1	0	The ALARM always is **1**.. (It can be used for reset latches and for other task when it is called in program or digital)	
0	0	0	0	0	0	1	1	Active while system error exists	
0	0	0	0	0	1	x		Active while error is on control block **x** (where **x**: 00→1, 01→2, 10→3, 11→4 the control block number)	
0	0	0	0	1	0	x		Active while autotune operates in control block **x** (where **x**: 00→1, 01→2, 10→3, 11→4 the control block number)	
0	0	0	0	1	1	x		Active while control block **x** is in manual mode (where x: 00→1, 01→2, 10→3, 11→4 the control block number)	
0	0	0	1	0	x			Active while relay output **rLbL** [**x**] = **1**. [wherel **x**: 000→1.,.. 001→2.,…111→8. (E1…E8)]	
0	0	0	1	1	x			Active while relay output **rLbH** [**x**] = **1** [wherel **x**: 000→1.,.. 001→2.,…111→8. (E1…E8)]	
1	0	0	0	0	0	0	0	Active while a programmer exists, if **ConF/PrG/dEF [10] ≠ 00**	
1	0	0	0	0	0	0	1	Active while the programmer is in **AUTOWAIT** state (AUTOWAIT= waiting to [PV=SP] state)	
1	0	0	0	0	0	1	0	Active while programmer is in **HOLD** state (HOLD = programmer is stopped)	
1	0	0	0	0	0	1	1	Active while programmer is in **NEXT** state, the period ~0,5 s (**Advn** = jumping ahead to the next segment by key)	
1	0	0	0	0	1	0	0	Active while programmer is running except during the start delay time	
1	0	0	0	0	1	0	1	Active while programmer is in start delay time segment	
1	0	0	0	0	1	1	0	Active while programmer executes a segment, the active period ~0,5 s	
1	0	0	0	0	1	1	1	Active while programmer is in the **End** segment, the active period ~0,5 s	
1	0	0	0	1	0	0	0	Active while the time base of the programmer is in **min/s** (minute/second)	
1	0	0	0	1	0	0	1	Active while the time base of the programmer is in **h/min** (hour/minute)	
1	0	0	0	1	0	1	0	Active while the time base of the programmer is in **d/h** (day/hour)	
1	0	0	0	1	0	1	1	Active while programmer executed a **soak** segment in control block **1**	
1	0	0	0	1	1	0	0	Active while programmer executes a **run+** segment in control block **1**	
1	0	0	0	1	1	0	1	Active while programmer executes a **run-** segment in control block **1**	
1	0	0	0	1	1	1	0	Active while programmer is in timing (delay or period) in sequencer mode	
1	0	0	0	1	1	1	1	Active while programmer executes a **FLAG** statement	
1	0	0	1	0	x			Active while an event code is valid in programmer, **EvnL** [**x**] where **x**: 000→1.,.. 001→2.,…111→8. (E1…E8)	
1	0	0	1	1	x			Active while an event code is valid in programmer, **EvnH** [**x**] where **x**: 000→1., 001→2.,…111→8. (E1…E8)	
3rd part of the **ConF./ALr*/dEF** table. (The **3rd** part is valid when **ConF./ALr*/SEt[6] = 1**)									
0	0	0	0	0	0	0	0	There is not such a function.	
				0	0	0	1	Active while **Stnd./LOAd < ALSP** (**ALr*** is active while the processor has free capacity referring to ALSP value)	
				0	0	1	0	Monitors if the **Stnd./Sont/dAY<AL*/ALSP.** When it is in inverse state and the **ALSP** time passed will be active.	
				0	0	1	1	Thereis not such a function.	
				0	1	0	0	Printer exists, it is configured.	
				0	1	0	1	There is error in the printer chart.	
				0	1	1	0	MMC data logger is configured.	
				0	1	1	1	There is error in the MMC data logger values.	
				1	0	0	0	First communication channel is working; there was message in the last 10 minutes period.	
				1	0	0	1	First communication channel is continuously receiving broadcast messages.	
				1	0	1	0	First communication channel is continuously sending broadcast messages.	
				1	0	1	1	There is not such a function.	

7	6	5	4	3	2	1	0	Continuation of the **3rd** part of the **ConF./ALr*/dEF** table.
				1	1	0	0	Second communication channel is working; there was message in the last 10 minutes period.
				1	1	0	1	Second communication channel is continuously receiving broadcast messages.
				1	1	1	0	Second communication channel is continuously sending broadcast messages.
				1	1	1	1	The **rEGL, rEGH** state of **ALr*= Stnd./dInP/rEGL**[*] or **ALr* = Stnd./dInP/ rEGH [*]** in MODBUS always , or in **Stnd.** page can be set if (**mmi2[1]=1**
0	0	0	1	0	x	x	x	Active while **Di x = 1**, the **ALARM*** changes its value 0→1 or 1→0 (where **x**: 000 →0, 001 → 2,…111 → 7 bits of digital input (**dInP**[7654_210])
0	0	0	1	1	x	x	x	Active when **Stnd./dInP/Ctr* < ALr*/ALSP** (compares Di/time counter value to **ALSP**)
1	0	0	0	0	0	0	0	Active while programmer is in SHADOW mode.
1	0	0	0	0	0	0	1	Active while programmer executes a **FrE*** segment, where * = 1, 2, 3 and 4 the control block number.
1	0	0	1	1	0	x	x	Active when **Stnd./PrG/rEG* < ALr*/ALSP** . (compares counter value to **ALSP**)

code

7	6	5	4	3	2	1	0	**ConF./ALr*/SEt**	**ConF./ALr***
					1	x	x	Active when control block * process value (**PV**) has reached first the set-point (**SP**), where * = 1, 2, 3 or 4 and **x**: 00→1, 01→2, 10→3, 11→4 the control block number)	
				1				Active when the controller is **ON** state, (in **OFF** state **ALr* = 0**)	
			1					Makes inverse of the source of ALARM (see; ALARM function block diagram)	
		0						Process-ratio value configured in **ConF./ALr*/dEF** will be calculated by 15 s period arithmetical mean	
		1						Process-ratio value configured in **ConF./ALr*/dEF** will be calculated by 100 s period arithmetical mean	
	0							On **AL.** page the **AL. / ALr* / ALhY** (hysteresis) can be seen	**ConF./ALr*/dEF** 2nd. part is valid
	1							On **AL.** page the **AL. / ALr* / ALhY** (hysteresis) can not be seen	**ConF./ALr*/dEF** 3rd. part is valid
1								On **AL.** page the **AL. / AL*/ ALSP** and **AL. / ALr* / ALhY** (ALARM SP and hysteresis) can not be seen	

for Events

7	6	5	4	3	2	1	0	**Conf./ALr*/LGE1** vagy **Conf./ALr*/LGE2**	**Conf/ALr***
0	0							There is not any Boole algebraic operation	Enabling with set; **Conf/SYSt/mmi4[0]=1** {ALr*} = [{ALr*} logical operation {2. operand}] Where **x**: 000→1.,.. 001→2.,…111→8.
0	1							XOR	
1	0							OR	
1	1							AND	
		1						Inverts the 2. operand	
			0	0		x		2. operand : digital input Stnd/dInP/dInP(x)	
			0	1		x		2. operand : alarm outputs Stnd/Alrb/Albl(x)	
			1	0		x		2. operand: event outputs Stnd/PrG/EvnL(x)	
			1	1		x		2. operand: event outputs Stnd/PrG/EvnH(x)	

7	6	5	4	3	2	1	0	**Conf./ALr*/LGA1** vagy **Conf./ALr*/LGA2**	**Conf/ALr***
0	0							There is not any Boole algebraic operation	Enabling with set; **Conf/SYSt/mmi4[0]=1** {ALr*} = [{ALr*} logical operation {2. operand}] Where **x**: 000→1.,.. 001→2.,…111→8.
0	1							XOR	
1	0							OR	
1	1							AND	
		1						Inverts the 2. operand	
			0	0		x		2. operand : ALARM output Stnd/ALrb/ALbL(x)	
			0	1		x		2. operand : ALARM output Stnd/ALrb/ALbH(x)	
			1	0		x		2. operand : ALARM output Stnd/ALrb/rLbL(x)	
			1	1		x		2. operand : ALARM output Stnd/ALrb/rLbH(x)	

7	6	5	4	3	2	1	0	**ConF./ALr*/dEcL**	**ConF./ALr***
		1						The red beacon for ALARM state does not function on the front panel	
	1							Inverts the red beacon for ALARM state. Inactiv → is lit.	
1								Inverse function for properties of **ConF./ALr*/dEF, ConF./ALr*/SEt, ConF./ALr*/dEcL**	
							1	The program overwrites the ConF./ALr*/ALSP with the value stored in FLAG ALSP*	
						1		The program overwrites the **ConF./ALr*/ALdt** and the **ConF./ALr*/AHdt** with the value stored in **AtL*** and **AtH***	

7	6	5	4	3	2	1	0	**ConF/ALr*/OPt**	**ConF./ALr***
						0	0	There is not any latch	
						0	1	ALARM latches by any type of signal leading or trailing edge 0 → 1, or 1 → 0	The latch saves the first change of signal and holds it to clearing
						1	0	ALARM latches by the first leading edge 0→1	
						1	1	ALARM latches by the first trailing edge 1→0	
				0	0			There is not latch clearing by ALARM	The **ALr*** latch will be cleared by the state of the relay **Stnd./ALrb/rLbH**: relay 12 11 10 / code 11 10 01
				0	1			ALARM* latch will be cleared if R10=1	
				1	0			ALARM* latch will be cleared if R11=1	
				1	1			ALARM* latch will be cleared if R12=1	
		0	0					The ALARM has not a timer function (time relay)	
		0	1					Repeat cycle timer starts when **ALR* = 1**, when its value is fixed to **0** it does not start	
		1	0					Restartable timer (the new input signal restarts the timer, in spite of the fact that the delay time did not elapse)	
		1	1					Not restartable timer (the new input signal can not start the timer, while the delay time did not elapse)	
	0							TIMER → LATCH in sequence	
	1							LATCH → TIMER in sequence	
#								Reserved for development	

7	6	5	4	3	2	1	0	**ConF/ALr*/SOFt**	**ConF./ALr***
0	0	0	0	0	0	0	0	R* = ALr*	where * is the relay and ALARM number y: 00 → 1, 01 → 2, 10 → 3, 11 → 4 x: 0000 → 1, 0001 → 2,…1111 → 16
0	0	0	1	x				R* = ALr* XOR ALrx	
0	0	1	0	x				R* = ALr* OR ALrx	
0	0	1	1	x				R* = ALr* AND ALrx	
0	1	y		x				R* = ALr* OR (ALrx OR ALry)	
1	0	y		x				R* = ALr* OR (ALrx AND ALry)	
1	1	y		x				R* = ALr* AND (ALrx OR ALry)	

Parameter value	**Conf./ALr***
ALdt	When ALr* changing (1 → 0) delay starts by this value [s] (initialized by trailing edge)
AHdt	When ALr* changing (0 → 1) delay starts by this value [s] (initialized by leading edge)
ALSP	Setpoint of ALr*. Can be set in **AL./ALr*/ALSP** too.
ALhY	Hysteresis of ALr*. Can be set in **AL./ALr*/ALhY** too.

3.5. Realtime clock configuration

7	6	5	4	3	2	1	0	**ConF./rtc/rtc**	**ConF./rtc**
1								The **Stnd./rtc** (real time clock) cannot be seen	
	1							The **Stnd./rtc** cannot be set	
		0						It retuns after clock error with last valid time. The right time must be set. Error code: **E.rtc**	
		1						In error code state only the code can be seen. The right time must be set. Error code: **E.rtc**	

Parameter value		**ConF./rtc**
sec	Adjusting and reading of the real time clock	second
h/m		hour : minute
dAY		day
mont		month
YEAr		year

3.6. Communucation configuration

7	6	5	4	3	2	1	0	**ConF./Com*/dEF**	**ConF./Com1**
					0	0	0	1200	Baud rate [bit/s]
					0	0	1	2400	
					0	1	0	4800	
					0	1	1	9600	
					1	0	0	19200	
					1	0	1	38400	
					1	1	0	There is not communication	
					1	1	1	There is not communication	
		0	0	0				ASCII: 8 bit 1 stop bit	
		0	0	1				ASCII: 7 bit 2 stop bit	
		0	1	0				ASCII: 7 bit even parity, 1 stop bit	
		0	1	1				ASCII: 7 bit odd parity, 1 stop bit	
		1	0	0				RTU: 8 bit 1 stop bit	
		1	0	1				RTU: 8 bit 2 stop bit	
		1	1	0				RTU: 8 bit even parity, 1 stop bit	
		1	1	1				RTU: 8 bit odd parity, 1 stop bit	
	1							Receives the broadcast messages (for all addresses) SLAVE	
1								Sends the broadcast messages (for all addresses) MASTER	

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**ConF./Com1/SEt**	**ConF./Com1**
							1	The MODBUS does not overwrite the menu items (read only operation)	
						1		The MODBUS does not overwrite the data of the programmer (read only operation)	
					1			The MODBUS does not overwrite the special menu items (read only operation)	
				1				If there is not a usable broadcast message for 10 s, the controller turns itself to MASTER operation. If this controller contains a programmer it ceases its SHADOW state (The SHADOW properties can be configured in programmer pages **ConF./PrG/OPt[3] = 1**)	

Parameter value	**ConF./Com1/Addr**
Addr	Communication (access) address of the controller: 0…255
oSP1	SP of the 1st control block is shifted to the SP message of the MASTER (control block) by this value It is valid for SLAVE controller only: **ConF./Com*/dEF[6] = 1**

4. CAL configuration

4.1. Cold junction configuration

7	6	5	4	3	2	1	0	**CAL./CJ/dEF**	**CAL./CJ**
						0	0	External cold junction KTY	
						0	1	External cold junction Pt100	
						1	0	Cold junction 0ºC, fixed	
						1	1	Cold junction 25ºC, fixed	
					1			Cold junction value appears on **Stnd.** page	
				1				The multichannel controller can be changed to single channel, in such case the cold junction will be on 14-16 terminals. CAL./ In2…In'8 du not exist.	
		0	0					The strongest filter for service measurement. (recommended)	
		0	1					Strongest filter for service measurement.	
		1	0					Weak filter for service measurement.	
		1	1					The weakest filter for service measurement.	
	0							Strong stochastic filter (recommended). For cold junction and system measurements.	
	1							Weak stochastic filter	
1								"There is not a configured control block (**In** *)" Inhibition of the A/D type error messages.	

Parameter value		**CAL./CJ**
ShFt	Shift of the temperature of the cold junction. (Note! the shift of the value of the sensor is at: **CAL./In*/ShFt** !!!)	

4.2. Input calibration

7	6	5	4	3	2	1	0	**CAL./In*/dEF** where * = 1, 2, 3, … 6		CAL./In*
0								Celsius temperature scale	For TC and RTD inputs When a whole to tenth change occurs the order of magnitude of the saved values are altered. So you must choose the order of magnitude during the configuration. After the configuration you must overwrite the values too.	
1								Fahrenheit temperature scale		
	0							Decimal Point: whole numbers only		
	1							Decimal Point: tenths of degrees		
0	0							* * * * there is not a decimal point	Valid for linear inputs (V, mA, Ohm)	
0	1							* * *.* one virtual decimal place		
1	0							* *.* * two virtual decimal place		
1	1							*.* * * three virtual decimal place		
		0	0	0	0	0	0	Input is disabled		
		0	0	0	0	0	1	M (Cu-Kopel) TC	-200.0 …100.0°C	TC input group
		0	0	0	0	1	0	T (Cu-CuNi) TC	-250.0…400.0 °C by IEC625-2	
		0	0	0	0	1	1	U (Cu-CuNi) TC	-250.0…400.0°C by DIN43710	
		0	0	0	1	0	0	J (Fe-CuNi) TC	-150 …1200°C by IEC625-2	
		0	0	0	1	0	1	L (Fe-CuNi) M TC	-150.0 …900.0°C by DIN 43710	
		0	0	0	1	1	0	E (NiCr-CuNi) TC	-150.0 …999.9°C by IEC625-2	
		0	0	0	1	1	1	N (NiCrSi-NiSiMg) TC	-200 …1350°C by IEC625-2	
		0	0	1	0	0	0	K (NiCr-NiAl) TC	-250 …1377°C by IEC625-2	
		0	0	1	0	0	1	P (Platinel) TC	0 - 1395 C °	
		0	0	1	0	1	0	S (Pt10Rh-Pt) TC	-50 …1800°C by IEC625-2	
		0	0	1	0	1	1	R (Pt13Rh-Pt) TC	-50 …1800°C by IEC625-2	
		0	0	1	1	0	0	B (Pt30Rh-Pt6Rh) TC	-50 …1830°C by IEC625-2	
		0	0	1	1	0	1	A (W5Rh-W26Rh) TC	-50 …2500°C	
		0	0	1	1	1	0	C (W5Rh-W26Rh) TC	-50 …2500°C	
		0	0	1	1	1	1	Ni/Ni18%Mo	0 - 1100°C	
		0	1	0	0	#	#	Reserved for development		Linear input group (Note: the potentiometric inputs have not error messages, because they are not interpretable)
		0	1	0	1	0	0	Potentiometer input 500 Ohm	ratio measurement (min 50, max 500)	
		0	1	0	1	0	1	Potentiometer input 5k Ohm	ratio measurement (min 0,5 max 5k)	
		0	1	0	1	1	0	Voltage input 50 mV		
		0	1	0	1	1	1	Voltage input 100 mV		
		0	1	1	0	0	0	Voltage input 200 mV	Current input 0…20 mA	
		0	1	1	0	0	1	Voltage input 40…200 mV	Current input 4…20 mA	
		0	1	1	0	1	0	Voltage input 1 V		
		0	1	1	0	1	1	Voltage input 0,2…1 V		
		0	1	1	1	0	0	Voltage input 2 V		
		0	1	1	1	0	1	Voltage input 0,4…2 V		
		0	1	1	1	1	0	Voltage input 0…5 V		
		0	1	1	1	1	1	Voltage input 1…5 V		
		1	w	0	0	0	0	Pt100 RTD	-250…850°C by DIN-IEC751	**w = [0]** 2-wire RTD group **w = [1]** 3-wire RTD group They act like a linear input, decimal point can be configured.
		1	w	0	0	0	1	Pt200 RTD	-250…850°C by DIN-IEC751	
		1	w	0	0	1	0	Pt500 RTD	-250…850°C by DIN-IEC751	
		1	w	0	0	1	1	Pt1000 RTD	-250…850°C by DIN-IEC751	
		1	w	0	1	0	0	JPt100 RTD	-250…850°C	
		1	w	0	1	0	1	JPt200 RTD	-250…850°C	
		1	w	0	1	1	0	JPt500 RTD	-250…850°C	
		1	w	0	1	1	1	JPt1000 RTD	-250…850°C	
		1	w	1	0	0	0	KTY83 Si thermistor	-55~175°C	
		1	w	1	0	0	1	Cu10 RTD	-50…180.0°C (base point = 25°C)	
		1	w	1	0	1	0	Cu100 RTD	-50…180.0°C	
		1	w	1	0	1	1	Ni100 RTD	-60…250.0°C	
		1	w	1	1	0	0	Ni120 RTD	-60…250.0°C	
		1	w	1	1	0	1	FeNi604 RTD	-200…240	
		1	w	1	1	1	0	Resistor input	0…500 Ohm	
		1	w	1	1	1	1	Resistor input	0…5 kOhm	

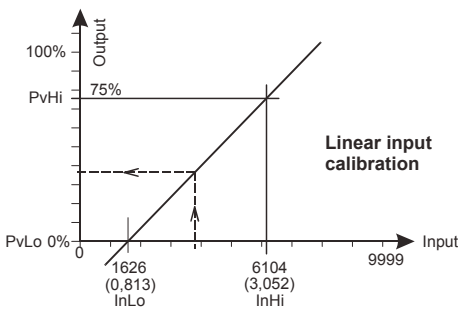
7	6	5	4	3	2	1	0	**CAL./In*/Unit** where * = 1, 2, 3, … 6	**CAL./Unit**
Enabling of the usage of units: **ConF./SYSt/mmi3[2]=1**									
0	0							4 s	Cycling time of appearance on the display (in seconds), The value and its unit alternate on the display
0	1							8 s	
1	0							16 s	
1	1							32 s	
		0						Appearing time on display 0,4 s	
		1						Appearing time on display 1 s	
			0	0	0	0	0		There is not any unit display
			0	0	0	0	1	ºC	Celsius degree
			0	0	0	1	0	ºF	FAHRENHEIT degree
			0	0	0	1	1	bar	pressure
			0	0	1	0	0	mbar	pressure
			0	0	1	0	1	PSI	pressure
			0	0	1	1	0	rh	relative humidity
			0	0	1	1	1	A	ampere
			0	1	0	0	0	mA	milliampere
			0	1	0	0	1	µA	microampere
			0	1	0	1	0	V	volt
			0	1	0	1	1	mV	millivolt
			0	1	1	0	0	µV	microvolt
			0	1	1	0	1	ohm	resistance
			0	1	1	1	0	rPm	revolution per minute
			0	1	1	1	1	PH	hydrogen-ion concentration
			1	0	0	0	0	mm	millimeter
			1	0	0	0	1	m	meter
			1	0	0	1	0	m3	cubic meter
			1	0	0	1	1	lit	liter
			1	0	1	0	0	O2	oxygen
			1	0	1	0	1	CO	carbon monoxide
			1	0	1	1	0	CO2	carbon dioxide
			1	0	1	1	1	%	per cent
			1	1	0	0	0	-1-	
			1	1	0	0	1	-2-	
			1	1	0	1	0	-3-	
			1	1	0	1	1	-4-	
			1	1	1	0	0	-5-	
			1	1	1	0	1	-6-	
			1	1	1	1	0	-7-	
			1	1	1	1	1	-8-	

7	6	5	4	3	2	1	0	**CAL./In*/FILt** where * = 1, 2, 3, … 6	**CAL./In***
1								Stochastic filter is on. It filters the stochastic big impulses. Recommended.	
	0	0	0	0	0	0	0	Input filter is off [its value = 0]	Can be set freely between 0-127 in binary form
	0	0	0	0	0	0	1	Weakest input filter [its value = 1]	
	0	0	0	0	1	1	1	Recommended weak input filter [its value = 7]]	
	0	0	0	1	1	1	1	Recommended intermediate input filter [its value = 15]	
	1	1	1	1	1	1	1	Strongest input filter [its value = 127] Against extreme big noises.	

7	6	5	4	3	2	1	0	**CAL./ In*/dECL** where * = 1, 2, 3, … 6	**CAL./In***
				0	0	0	0	nincs	Here may be set the fixing function (**Stnd./ALrb/rLb***) of the value of the input on the display. It breaks off the measurement. (Note! it is not a correct state because **Stnd./ALrb/rLb***=constant and SP= constant. So the error messages are not correct. Use in special purpose only and analyse it by experiments. Example: in case of **In2/dECL[3210]=0101** if **Stnd./ALrb/rLb[4]=1** will be accomplished, the **Stnd./In2** freezes with its last value. After **Stnd./ALrb/rLb[4]=1** changes to **0** the controller will continue with this value. (See: ALARM function block diagram) Note. The **Stnd./ALrb/rLbL** and the **Stnd./ALrb/rLbH** are values of binary ALARM function. These are continuously counted by the controller. So if the 1 No relay is automatically reserved for a control block, in spite of that it can be used for other task e.g. for freezing the measured value.
				0	0	0	1	**Stnd/ALrb/rLbL [0]**	
				0	0	1	0	**Stnd/ALrb/rLbL [1]**	
				0	0	1	1	**Stnd/ALrb/rLbL [2]**	
				0	1	0	0	**Stnd/ALrb/rLbL [3]**	
				0	1	0	1	**Stnd/ALrb/rLbL [4]**	
				0	1	1	0	**Stnd/ALrb/rLbL [5]**	
				0	1	1	1	**Stnd/ALrb/rLbL [6]**	
				1	0	0	0	**Stnd/ALrb/rLbL [7]**	
				1	0	0	1	**Stnd/ALrb/rLbH[0]**	
				1	0	1	0	**Stnd/ALrb/rLbH[1]**	
				1	0	1	1	**Stnd/ALrb/rLbH[2]**	
				1	1	0	0	**Stnd/ALrb/rLbH[3]**	
				1	1	0	1	**Stnd/ALrb/rLbH[4]**	
				1	1	1	0	**Stnd/ALrb/rLbH[5]**	
				1	1	1	1	**Stnd/ALrb/rLbH[6]**	
			1					The Gain like correction can be set in **(CAL./In*/uGn)** for **In*** input	
0								Error message pops up after 5 s and will be effective	
1								Error message pops up after 30 s and will be effective	
	0							The message will be cleared automatically when the error shall have been repaired	
	1							The message will be cleared by a reset on mains or by **ConF./SYSt/dFLt=128** after the error shall have been repaired	
		0						The error message conforms with the type of error (e.g. control block error or system error)	
		1						All type of error messages will appear and act as a system error	

7	6	5	4	3	2	1	0	**CAL./ In*/ mAth** where * = 1, 2, 3, … 6	**CAL./In***
				0				Adds to the **In*** another **In*** or **In'7** which is selected by the [210] bits	**In*** can not make an algebraic operation with itself
				1				Subtracts from the **In*** another **In*** or **In'7** which is selected by the [210] bits	
					¤	¤	¤	[¤¤¤] the binary code of the "another" **In*** or **In'7,** [000] = there is not an "another" **In***, [001] = **In1** … [111] = **In'7**	
0								The algebraic operation result will not be modified	
1								The algebraic operation result will be divided by 2	
	u	u	u					[**uuu**] The binary code of the user table [000] = is not, [001] = **first**,… [111] = **seventh** The tables are in the page: **utb*.**	

Parameter value			**CAL./In***
ShFt	Offset of the displayed PV value in its unit. + value shifts upwards, - value downwards.		
InLo	The lower value of the linear input in A/D unit	The range of the A/D unit is 0-9999. The points must be assigned proportionally. E.g. the 2500 value is the quarter of the range. If **InLo = InHi** than the whole range will be valid automatically.	
InHi	The upper value of the linear input in A/D unit		
PvLo	Low value of range	Linear input calibration values	If **PvLo** = **PvHi** the common linear inputs are valid (0/4-20 mA, 0-1 V)
PvHi	High value of range		



Example

Date of HIH-3605-A-CP type RH sensor: 0,813 V → 0% 3,052 V → 75%
Linear input range: 0 … 5 V
From these: 0 V → 0 5 V → 9999 A/D converter value
proportionally: 0,813 V → 1626 3,052 V → 6104 A/D converter value

InLo and **InHi** overwrite the common linear inputs (mA, mV, Ohm)

uGn	The Gain like correction can be set between 000.1 and 999.9. Between 100.0 and 000.0 does not make any change. Above 100.0 increases under it decreases the sign of the sensor. The correction equals with the sensors error. So at +1,5% error 098.5 value has to set. Enabling by **ConF./SYSt/mmi5[65]** and **CAL./In*/dECL[4]**.

The decimal point

The controller operates the decimal point by mathematical rules, when TC or RTD is configured. But take it account that after changing the resolution (0.1 ↔ 1.0) all the temperature values will change too, e.g. 345 will be 34.5. So after such type of change you must rewrite the parameter values (SP, SPLo, SPHi, SP/mSP* etc).

The actuator output Y is always with one decimal, e.g. 45.7. It do not depend on resolution.

The virtual decimal point is only a sign. The displayed number is a whole number and the results are always whole numbers too. You ought to configure the place of the decimal point at the end of the configuration process. The virtual decimal point helps the user, when the display is read.

4.3. Mathematical input calibration

7	6	5	4	3	2	1	0	**CAL./ In' */dEF** wherel * = 1 and 2	**CAL./ In' ***
0	0							* * * * there is not decimal point	
0	1							* * *.* 1 decimal place (with virtual decimal point)	
1	0							* *.* * 2 decimal places (with virtual decimal point)	
1	1							*.* * * 3 decimal places (with virtual decimal point)	
		1						Clearing the error of mathematical channel. It is a special thing after when this was intentionally driven in error state. It is not an operating condition.	
			#					Reserved for development	
				0	0	0	0	There is not a deriving (mathematical channel) input, its value = 0	
				0	0	0	1	Stnd./In1	x axis is the first input of **In'***, where *=7 and 8. The first and second inputs must be different. Examples and descriptions are in: “Specific applications”.
				0	0	1	0	Stnd./In2	
				0	0	1	1	Stnd./In3	
				0	1	0	0	Stnd./In4	
				0	1	0	1	Stnd./In5	
				0	1	1	0	Stnd./In6	
				0	1	1	1	Stnd./In' 7	
				1	0	0	0	SP1	SP1 … SP4 are the calculated set-points of the control bocks which are seen on the green display. See flowchart **"The configuration of the control block"**
				1	0	0	1	SP2	
				1	0	1	0	SP3	
				1	0	1	1	SP4	
				1	1	0	0	Stnd./Y1	Y1 …Y4 are the values of actuator outputs of control blocks
				1	1	0	1	Stnd./Y2	
				1	1	1	0	Stnd./Y3	
				1	1	1	1	Stnd./Y4	

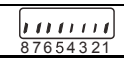
7	6	5	4	3	2	1	0	**CAL./In'*/Unit** where * = 1 and 2	**CAL./ In' ***
Agrees exactly with table **CAL./In*/Unit**									

7	6	5	4	3	2	1	0	**CAL./In'*/FILt** where * = 1 and 2	**CAL./ In' ***
Agrees exactly with table **CAL./In*/FILt**									

7	6	5	4	3	2	1	0	**CAL./In'*/dEcL** where * = 1 and 2	**CAL./ In' ***
Agrees exactly with table **CAL./In*/dEcL**									

7	6	5	4	3	2	1	0	**CAL./ In'*/ mAth** where * = 1 and 2	**CAL./ In' ***
Agrees exactly with table **CAL./In*/mAth**									

Parameter value	where *=7 and 8	**CAL./In'***
ShFt	Offset of the displayed PV value in its unit. + value shifts upwards, - value downwards.	
InLo	The lower value of the linear input in A/D unit	(**In' 7** and **In' 8** inner auxiliary inputs derive their signals from **In*, In' 7**, **SP***, **Y*** values. These can be seen on the **Stnd.** page. They can be configured by the bits of **CAL./In' */dEF[3210]**. E.g. one inner auxiliary input value may be the calculated value of the 3rd TC. The value of the output has not dimension and can be modified by user table too. So you can get an output in function of input. The operation can be seen in **CAL./In* “**Parameter value”.
InHi	The upper value of the linear input in A/D unit	
PvLo	Low value of PV range	
PvHi	High value of PV range	

7	6	5	4	3	2	1	0	**CAL./ In'*/mtrII** where * = 7 and 8	**CAL./ In'***
			0					Selection of the **mtr1.** (three dimensional) linearization table	
			1					Selection of the **mtr2.** (three dimensional) linearization table	
*	*	*						Selection of the **mtr3.** (three dimensional) linearization table. It will be valid if **Stnd./ALrb/rLbH[***]=1**, where: [***]=001→1, [***]=010→2, [***]=011→3, [***]=100→4, .. [***]=111→7	 87654321

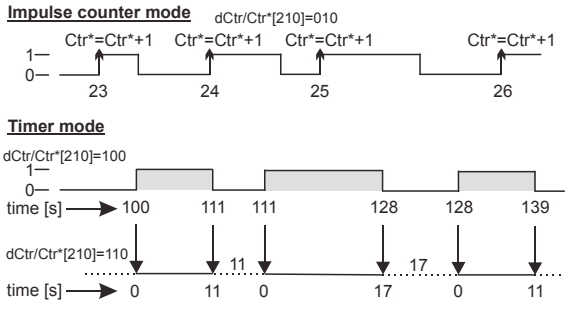
				0	0	0	0	Off state		y axis is the first input of **In'***, where *=7 and 8. The first and second inputs must be different. Examples and descriptions are in: "Specific applications".
				0	0	0	1	Stnd./In1		
				0	0	1	0	Stnd./In2		
				0	0	1	1	Stnd./In3		
				0	1	0	0	Stnd./In4		
				0	1	0	1	Stnd./In5		
				0	1	1	0	Stnd./In6		
				0	1	1	1	Stnd./In' 7		
				1	0	0	0	SP1	SP1 … SP4 are the calculated setpoints of the control block which can be seen on the green display.	
				1	0	0	1	SP2		
				1	0	1	0	SP3		
				1	0	1	1	SP4		
				1	1	0	0	Stnd./Y1	Y1 …Y4 the actuator signals of the control blocks.	
				1	1	0	1	Stnd./Y2		
				1	1	1	0	Stnd./Y3		
				1	1	1	1	Stnd./Y4		

5. Linear output configuration

7	6	5	4	3	2	1	0	**CAL./Lin*/dEF** where * = 1 and 2	**CAL./Lin***
0								4 … 20 mA , 1 … 5 V, 2 … 10 V output	
1								0 … 20 mA , 0 … 5 V, 0 … 10 V output	
	0	0						Linear output is always operating.	
	0	1						If **ALARM F** is **on**, the value of the linear output is always 0 or 4 mA (liner output is in its base position).	
	1	0						If controller is **OFF**, the value of the linear output is always 0 or 4 mA (liner output is in its base position).	
	1	1						If **ALARM F** is **on**, the value of the linear output is always 0 or 4 mA (liner output is in its base position).	
			0	0	x	x	x	**In *** input will be (see on **Stnd.** page) on the linear output, where x = 000 → 1, 001 → 2 … 111 → 8	
			0	1	0	x	x	**SP*** calculated SP will be (see on the green display) on the linear output, where, x = 00 → 1, 01 → 2 … 11 → 4	
			0	1	1	x	x	**Y*** the actuator signal of the control block will be on the linear output, where x = 00 → 1, 01 → 2 … 11 → 4	
			1	0	x	x	x	If **rLbL**(x) = **1** (can be seen on **Stnd.** page) the value of the linear output is 20 mA, where x = 000 → 1, 001 → 2 … 111 → 8	For Relay, or SSd actuation (SSd: **S**olid **S**tate relay **d**river)
			1	1	x	x	x	If **rLbH**(x) = **1** (can be seen on **Stnd.** page) the value of the linear output is 20 mA, where x = 000 → 9, 001 → 10 … 111 → 16	

Parameter value		**CAL./Lin***
CAL./Lin*/LiLo	Low value of linear output range (this value is related to one of these points 0 mA, or 4 mA, or 1 V, or 2 V)	
CAL./Lin*/LiHi	High value of linear output range (this value is related to one of these points 20 mA, or 5 V, or 10 V)	

6. Imputs counter configuration

7	6	5	4	3	2	1	0	**CAL./dCtr/Ctr*** where * = 1,2, ... 8	**CAL./dCtr**
					0	0	0	There is not an impulse counter or timer function assigned to the digital inputs	
					0	0	1	Counts all leading and trailing edges	
					0	1	0	Counts the leading edges only	
					0	1	1	Counts the trailing edges only	
					1	0	0	Measures the **Stnd./dInP/dInp* = 1** status in second	
					1	0	1	Measures the **Stnd./dInP/dInp* = 0** status in seconds	
					1	1	0	Measures time from **1** status in seconds	
					1	1	1	Measures time from **0** status in seconds	
				1				Clears when the mains turns on	
			1					Clears the impulse counter and timer values in **OFF** state	
		1						Clears the impulse counter and timer values when **Stnd./ALrb/rLbL[0] = 1**	
	1							Clears the impulse counter and timer values when **Stnd./ALrb/rLbL[*] = 1**	
1								The impulse counter values can be seen on **Stnd./dInP/Ctr***	

7. Data acquisition memory card configuration

7	6	5	4	3	2	1	0	**CAL./FiLE/dEF**		**CAL./FiLE**
				0	0	0	0	There is not data acquisition		
				0	0	0	1		1	s (second)
				0	0	1	0		2	
				0	0	1	1		5	
				0	1	0	0		10	
				0	1	0	1		20	
				0	1	1	0		30	
				0	1	1	1		1	m (minute)
				1	0	0	0	Sample time. The time between two samples.	2	
				1	0	0	1	(The second sample time can be set in **CAL./FiLE/SEt)**	5	
				1	0	1	0		10	
				1	0	1	1		30	
				1	1	0	0		1	h (hour)
				1	1	0	1		2	
				1	1	1	0		6	
				1	1	1	1		12	
		0	0					In the menu configured samples are always being sent to the card.		
		0	1					Data acquisition operates only in **ON** state.		
		1	0					Data acquisition operates only if **rLbH[6]=1** (ALARM G is active)		
		1	1					Data acquisition operates only if **rLbH[7]=1** (ALARM F is active)		
	1							Receives the data of ALARM functions. (Can be query in **Stnd./ALrb/ALb***)		
1								Receives the data of RELAY states. (Can be query in **Stnd./ALrb/rLb***)		

7	6	5	4	3	2	1	0	**CAL./FiLE/SEt**	**CAL./FiLE**
				0	0	0	0	Function disabled	
				*	*	*	*	This is the second sample time. Code is the same as the **CAL./FiLE/dEF[3210]**	
0	x	x	x					If **Stnd./ALrb/rLbL**[x] = **1** the controller sends data with the second sample time, where x = 000 → 0, 001 → 1 … 111 → 7	
1	x	x	x					If **Stnd./ALrb/rLbH**[x] = **1** the controller sends data with the second sample time, where x = 000 → 0, 001 → 1 … 111 → 7	

7	6	5	4	3	2	1	0	**CAL./FiLE/rEG1**	**CAL./FiLE**
							1	In1	Sends the here selected configured control channel parameters. If the channel is not configured e.g. **CAL./In3/dEF[76543210]=00000000**, it does not send the **In3** in spite of **CAL./FiLE/rEG1[3]=1**
						1		In2	
					1			In3	
				1				In4	
			1					In5	
		1						In6	
	1							In7’	
1								In8’	

7	6	5	4	3	2	1	0	**CAL./FiLE/rEG2**	**CAL./FiLE**
							1	SP1	Sends the here selected configured control channel parameters. If the channel is not configured e.g. **CAL./In3/dEF[76543210]=00000000**, it does not send the **SP3** and **Y3** in spite of **CAL./FiLE/rEG1[62]=11.**
						1		SP2	
					1			SP3	
				1				SP4	
			1					Y1	
		1						Y2	
	1							Y3	
1								Y4	

7	6	5	4	3	2	1	0	**CAL./FiLE./OPt**	**FiLE./OPt**
						#	#	Reserved for development	
					1			The normal data acquisition (**CAL./FiLE/dEF**)is disabled.	
				1				This operation (**FiLE./OPt**) is enabled.	
0	x	x	x					When **Stnd./ALrb/rLbL**[x] = (0 ↔ 1) ALARM changes, the assigned data in **FiLE/rEG1** and **FiLE/rEG2** will be stored once on the card together with real time. Where x = 000 → 0, 001 → 1 … 111 → 7	
1	x	x	x					When **Stnd./ALrb/rLbH**[x] = (0 ↔ 1) ALARM changes, the assigned data in **FiLE/rEG1** and **FiLE/rEG2** will be stored once on the card together with real time. Where x = 000 → 0, 001 → 1 … 111 → 7	

8. Loggin and printing

8.1. 6 colour 32-channel hybrid data logger and chart recorder.

The KD9 universal compact controller contains one data logger and chart recorder.

The printer interface built in the KD9 prints configurable 210 mm or 345 mm wide chart on dot matrix printer. The controller and the printer are connected by parallel cable.
It prints continuous fanfold stock on clear paper up to one original and two copies. So it usable as a document for all of quality management system

There are thee field on the paper:

1. values of inputs, setpoints (temperature, pressure, etc.) lines on paper printed by A4 printer with 159 mm and by A3 printer 305 mm wideness.
2. the actuators %. in 20 mm wideness.
3. The ALARM states in 20 mm wideness configured in (**Stnd./ALrb/rLb***).

All of the printed values can be queried in the printer make them visible.

Technical data:

Channels:

6 input (analogue) channel for measured data.	15 types, TC, types RTD, 1 type, KTY83 thermistor, 2 types potentiometer, types, resistor, current 0/4-20 mA, 10 types voltage up to10 V, 9 types linearization table.
6 input (analogue) channel for setpoint if it registers control loops.	It is well usable for SP programmer
4 input (analogue) channel for output actuator (Y), if it registers control loops.	It is well usable for PID tuned control loops, where the actuator is varying from 0 to100%, or from -100 to +100 %.
16 output	output of relay, or TTL states

Useful paper width depending on the type of printer: 210 mm or 345 mm.

Chart speed: 1,2-1291 mm/hr, adjustable in 11 steps.

A/D converter resolution: 120000 for every measuring range.

Analogue inputs: cold junction compensation for TC and three wire connection compensation for RTD.

Sample time: from 1 to 1080, adjustable in 11 steps.

Printing accuracy: better then ±0,2 mm. (depends on the good condition of the printer mechanic)

Printing format: analogue graphs (curves), text in lines, digital printing for ALARM states

KD9 display: 3 lines 7 segments 4 digital number and mnemonic and 20 pictograms

Data on display: measured data, setpoint, actuator, program data in programmer, ALARM states.

Data logger: real time data acquisition on multimedia card. HAGA Display program (download www.hagamat.hu).

ALARM outputs: 11 relay or OPC + 5 TTL.

ALARM types: common types + complete time relays with latch.

Latch clearing: from program, with ALARM, with push button, with code.

Error sign: display blinking and sound.

Security: multi level, password

Marker: the chart can contain it by many type of event.

The KD9 can actuate parallel with a chart recorder and an MMC data logger so it can control big systems.

The structure of chart

The **1. field** can contain 12 sign. Colours are assigned to the signs if colour mode is configured.

The colours by channels: 1 and 7 **black**, 2 and 8 **magenta**, 3 and 9 **blue**, 4 and A **green**, 5 and b **violet**, 6 and C **orange**. The **Y** (actuator) signs are the same. The colours can be seen in the heading.

The parameter surface can be divided on stripes. Every stripe operates as a separate chart. The number of stripes may be 1, 2, 3, and 4. Every sign can be in any stripe. The same sign can be in more stripes. However, the maximal number of signs is 12 altogether.

Form of printing

The recorder prints points. The resolution may be three times bigger depending of data quantity. The chart lines will be more continuous.

When the data quantity is the biggest (3 times resolution and colour printing) then the recorder prints with 45 s sample time so 1 line (14,06 mm) needs 30 minutes. The recorder prints (1 times resolution and colour printing) in turbo mode with 3 s sample time that is 2 minutes/line. The data refer to a printer with 200 character/s print speed.

It is very advisable to use Uninterruptible Power Source (UPS) if you want important data register without mistake.

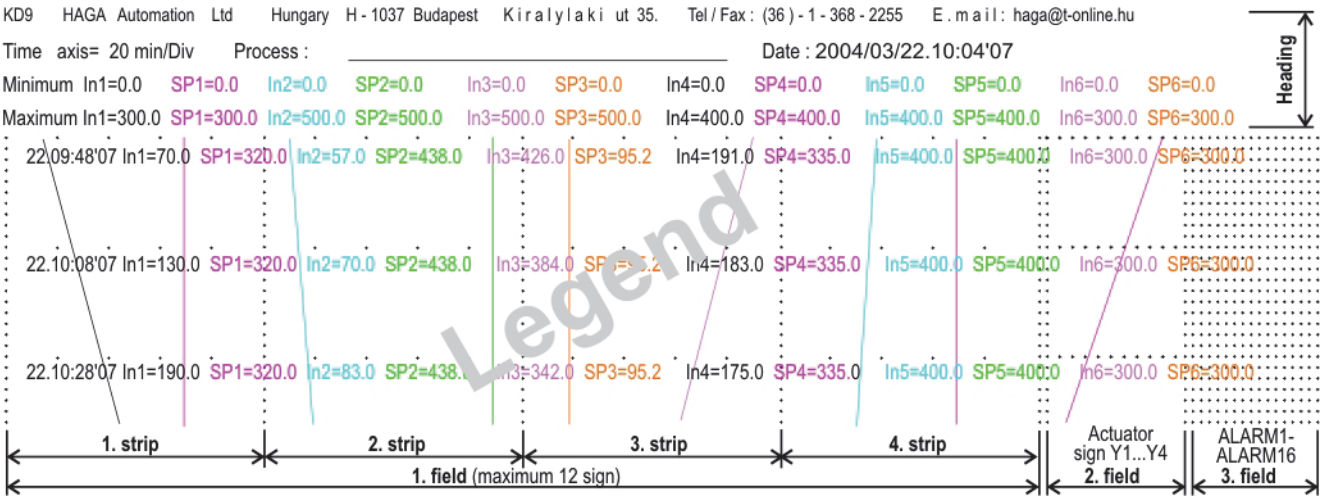
The KD9 can work with EPSON, STAR and PANASONIC matrix printer.

The time axis of the chart.

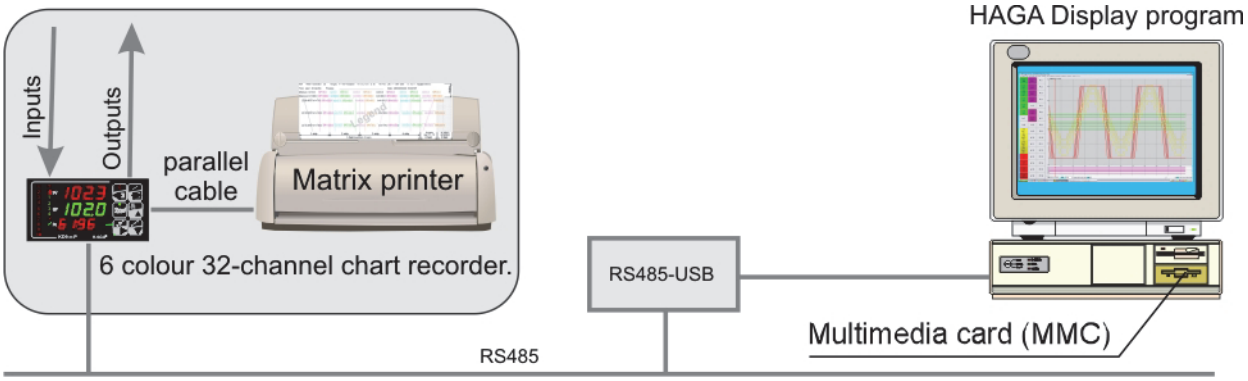
The recorder registers by relative or real time depending on the configuration. The inner clock works for about 2 weeks without power.

The time axis can be adjusted. The value of the real time clock appears in the chart heading in form: **year/month/day.hour:minute'second** and in the chart **day.hour:minute'second.**

The form of the chart



Communication properties of KD9



8.2. Chart recording on printer

When **Prnt./dEF[4] = 1** the actuators values (Y) appear in the 2. field automatically in -100.0 and 100.0, or 0.0 and 100.0 format depending on the configuration. However Y can be drawn in any strip giving its start and end values.

Prnt./dECL[**0**] = **1** means that the printer can work in turbo mode. In this mode, the printer cannot use the three-time speed. This function is automatically prints in the printer at once 8 data (namely 8-point line) because of that a mains failure causes some data loss.

If **Prnt./dECL**[**1**] = **1** it prints in 357 mm width. The contents of the chart do not alter only the x scale double.

8.3. The Printers

Switch off the NLQ (Near Letter Quality) option because it can cause line offset after the text print. The KD9 sends relevant command but some printer type cannot accept or this function is disabled.

The printer draws more continuous lines if three times resolution is valid. This works well with 9 pins printer. Other printers can draw ragged chart.

The printers do not contain the same command set. The KD9 can actuate many of them. These printers, that do not work well, are not "ESC/P" compatible. This must configure in the printer if it exists not in the controller.

The matrix printers are relatively slow. We do not know commonly the print speed. The program of KD9 is optimized for speed. Obvious that in case of black printing is not speed problem but in coloured mode, the data speed increases. The controller senses the data loss and prints a new heading. This abnormal heading shows the data overflow. Decrease the number of parameters or increase the time scale in **Prnt./dEF**. When the data quantity is the biggest (3 times resolution and colour printing) then the recorder prints with 45 s sample time so 1 line (14,06 mm) needs 30 minutes. The recorder prints (1 times resolution and colour printing) in turbo mode with 3 s sample time that is 2 minutes/line. The data refer to a printer with 200 character/s print speed.

Configure the KD9 to the printer by the next tables. Keep at the order of table. The configuration is possible in any state of the controller if it is enabled.

8.4. Priter configuration

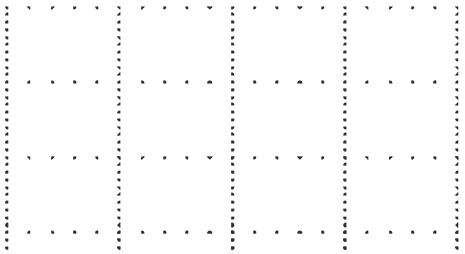
7	6	5	4	3	2	1	0	**Prnt./dEF**				**Prnt.**
								Sample time		Time for one page printing (A4)		Time scale		—
There is not					0	0	0	normal	turbo	normal	turbo	normal	turbo	—
					0	0	1	15 s	1 s	3,6 hr	14.3 min	10 min	40 s	The time scale is the period of printing a line. Writes the time ahead of every line. So if [210]=101 writes a line during 2 hr. Before the lines are the time e.g. : 2:00 4:00 6:00 ....etc.
					0	1	0	30 s	2 s	7,2 hr	28,7 min	20 min	80 s	
					0	1	1	45 s	3 s	10,8 hr	43,0 min	30 min	2 min	
					1	0	0	1,5 min	6 s	21,6 hr	1 hr 26 min	1 hr	4. min	
					1	0	1	3 min	15 s	1 day 19,1 hr	3,6 hr	2 hr	10. min	
					1	1	0	9 min	30 s	5 day 9,4 hr	7,2 hr	6 hr	20. min	
					1	1	1	18 min	45s	10 day 18,7 hr	10,8 hr	12 hr	30. min	
				1				Writes the values of variables with letters and numbers if 0 than does not print.						
			1					Draws the values of the actuator (**Y***) in the **2. field**. Sets automatically the scale. ♣☼						
		1						Draws the values of ALARM (**rLbL** and **rLbH)** in the 3**. field**. Order of appearances: 123456789AbCdEFG						
	1							Prints text only, does not draw the curves.						
1								Coloured printing. ♣☼						

Prnt./SEt

7	6	5	4	3	2	1	0	**Prnt./SEt**			**Prnt.**
					0	0	0	1 x 10	1 x 10	1x20	**Background lines** Calibration of horizontal (x) axis. Can be select the number of main and auxiliary lines in the area domain of parameter values. The main and auxiliary lines consist of points. The main lines have 10 points in a time scale the auxiliaries 1.
					0	0	1	2 x 10	2 x 10	2x10	
					0	1	0	--------	3 x 10	3x10	
					0	1	1	4 x 5	4 x 5	4x10	
					1	0	0	5 x 10	5 x 10	6x10	
					1	0	1	--------	6 x 10	9x10	
					1	1	0	10 x 1	10 x 10	12x10	
					1	1	1	16 x 1	15 x 10	16x10	

ribbon printing 57 mm | normal printing (A4) | Wide printing (A3)

Example: Prnt./SEt[210 = 011]



				1				Three-time dense printing for curve graph. (**1. field**) ♣☼♣
		0	0					Always is printing
		0	1					Prints in on state only.
		1	0					Prints if **Stnd./Alrb/rLbH[6]** = 1 (15. ALARM **ALrF**)
		1	1					Prints if **Stnd./Alrb/rLbH[7]** = 1 (16. ALARM **ALrG**)
	1							Clears the value of time axis after restart, starts at 00.00.
1								Does not write heading, starts the printing from stop state. Does not print in stop state (without data logging).

Prnt./dECL

7	6	5	4	3	2	1	0	**Prnt./dECL**	**Prnt./dECL**
							1	Turbo printing. Time scale by **Prnt./dEF**[**210**]. (Prints 8 point line at once)	♣☼♣
						1		Prints on wide printer (A3)	♣☼♣
					1			Writes seconds (s) in time value too.	
				1				It prints on ribbon printer (57 mm wide *CITIZEN CBM-920 4OPF compatible*).	☼
			1					It prints on more line at every change of **ALrb** (**Stnd./ALrb/rLbH[2]**) if **Prnt./dEf[6]=1** in text mode.	
		1						Printed appearance of ALARM state: , if [5] = 0 than:	♣☼♣
	1							Prints always in relative time, if [**6**]=**0** in real time by the inner clock.	
1								It does not print the data of the owner.	

Prnt./OPt

7	6	5	4	3	2	1	0	**Prnt./OPt**	**Prnt./OPt**
					0	0	0	There is not a second printing speed.	
					x	x	x	May be set a second printing speed by the x values in **Prnt./dEF=[210]**. It does not work in turbo mode, when **Prnt./dECL[0]=0.**	
0	x	x	x					Sends data with the second sampling time if **Stnd./ALrb/rLbL**[x] = **1** where x = 000 → 0, 001 → 1 … 111 → 7	
1	x	x	x					Sends data with the second sampling time if **Stnd./ALrb/rLbH**[x] = **1** where x = 000 → 0, 001 → 1 … 111 → 7	

Sign			Sign	
♣☼	**Prnt./dEF[7--4]** effect will be modified	**0--0** Table like printing in columns. **0--1** There is not in column fixed arrangement, prints data by one space. **1--0** Prints ordinal numbers and data only. **1--1** prints data only	☼	Selection of ***CITIZEN CBM-920***. The **Prnt.** configurational tables change by the here visible signs.
♣☼♣	These setting are not valid for 57 mm wide *CITIZEN CBM-920* printer.			

Prnt./rEG

7	6	5	4	3	2	1	0	**Prnt./rEG¤/dEF** where ¤ = 1, ... 9, A, b, C (2x6 colour, or channel) ☼	**Prnt./rEG¤**
				0	x	x	x	**In** (1 ... 8) values of inputs (seen in the **Stnd.**) in the ¤ channel, where ¤ = 000 → 1, 001 → 2 … 111 → 8	
				1	0	x	x	**SP** (1 ... 4) counted SP (seen in the green display) in the ¤ channel, where ¤ = 00 → 1, 01 → 2 … 11 → 4	
				1	1	x	x	**Y** (1 ... 4) the actuator value of control channel in the ¤ channel, where ¤ = 00 → 1, 01 → 2 … 11 → 4	
		0	0					Positioning the selected parameter in the **1. strip**.	
		0	1					Positioning the selected parameter in the **2. strip**.	
		1	0					Positioning the selected parameter in the **3. strip**.	♣☼♣
		1	1					Positioning the selected parameter in the **4. strip**.	♣☼♣
0	0							The **1. field** divided to **1** strip.	
0	1							The **1. field** divided to **2** strip.	
1	0							The **1. field** divided to **3** strip.	♣☼♣
1	1							The **1. field** divided to **4** strip.	♣☼♣

Parameter value	where ¤ = 1, ... 9, A, b, C (2x6 colour, or channel) ☼	**Prnt./rEG¤**
Prnt./rEG*¤/PrLo	The value of the selected parameter at the start point.	The actuator (Y) selects automatically its start and end point only in **1. field.**
Prnt./rEG*¤/PrHi	The value of the selected parameter at the end point.	

☼ ***CITIZEN CBM-920*** in case of usage **Prnt./rEG¤/dEF,** where ¤ = 1, ... 4

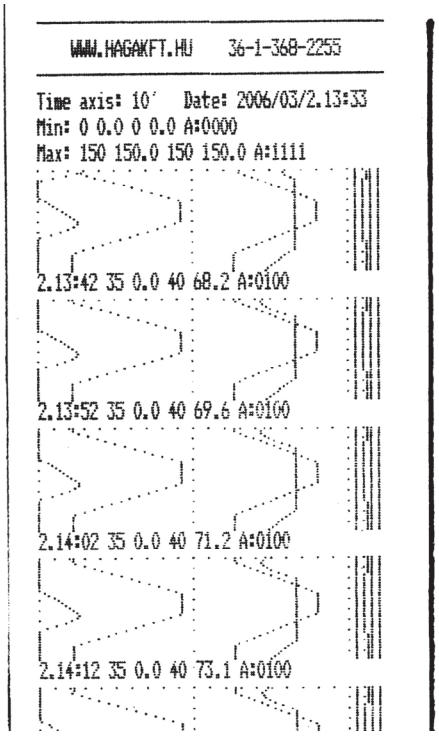
The ribbon printer

The A3 and A4 printer can register a lot of data. These printers need quite big places and are sensible to contamination. In such places where only some data have to register can be use 57 mm wide ribbon printer.

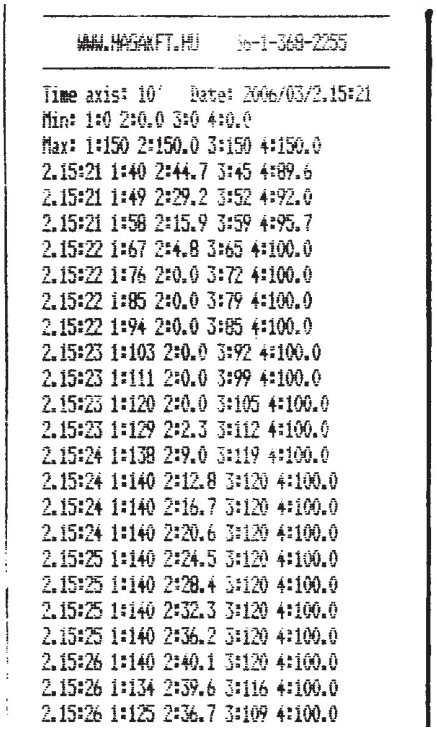
The ***CITIZEN CBM-920*** printer is able for printing characters and graphs. The KD9 sends the real time data so a usable diagram is drawn. The code set is incompatible with big printers so the KD9 cannot work without alteration. The incompatibility to and fro valid.

Panel mounting, the cutout: 109 $_{-0,5}$ x 62 $_{-05}$.

Printed fields



Printed chart



Printed text

9. Linearisation tables

9.1. utb.*

Parameter value	where * = 1, 2, 3, … 7	utb*.
utb*./^{co} *n*	The ***n***th coordinate on x axis, where * = 1 … 7 and ***n*** = 1… 32	
utb*./^{co} *n*/^{co}in	The x value of ***n***th coordinate on x axis (distance from origin with sign) This value must be monotone growing: x(***n***)<x(***n***+1)	
utb*./^{co} *n*/^{co}ou	The y value of ***n***th coordinate on y axis (distance from origin with sign)	

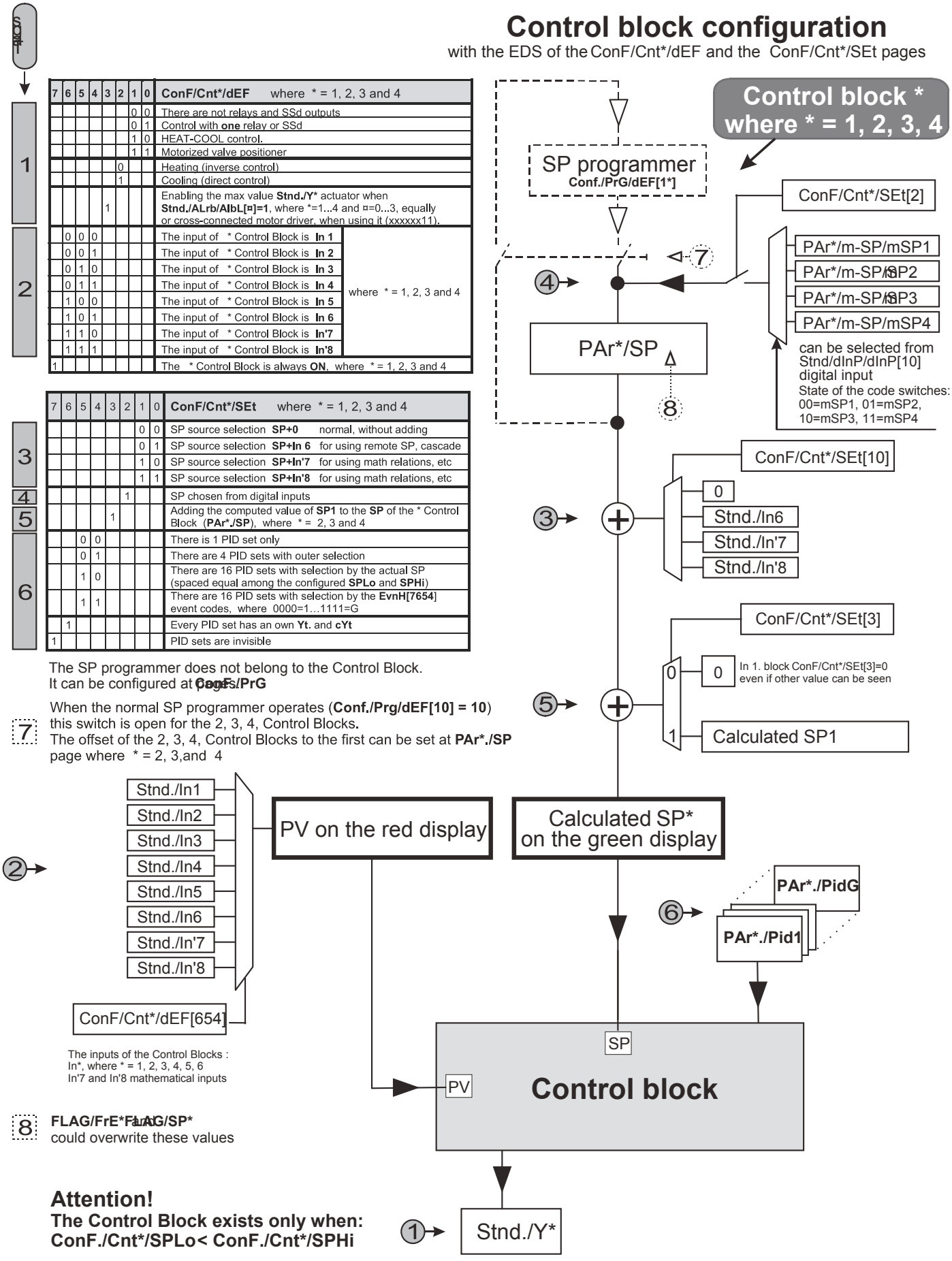
9.2. mtr.*

Parameter value	where * = 1 and 2	mtr*.
mtr*./r.^{coo}/rov.*n*.	The x value of ***n***th coordinate on x axis, where * = 1 … 2 and **n** = 1… 9, A … G (distance from origin with sign). This value must be monotone growing: x(***n***)<x(***n***+1)	
mtr*./c.^{coo}/coL.*m*.	The y value of ***m***th coordinate on y axis, where * = 1 … 2 and ***m*** = 1… 9, A … G (distance from origin with sign). This value must be monotone growing: y(***m***)<y(***m***+1)	
mtr*./1.rov/ *m*.coL ↓ **mtr*./*n*.rov/ *m*.coL** ↓ **mtr*./G.rov/ *m*.coL**	The z value of ***nm***th coordinate on z axis, where * = 1 … 2 and ***n*** = 1… 9, A … G and ***m*** = 1… 9, A … G (distance from origin with sign). **It is expedient**: begin with giving the ***m*.coL** value to the first row, than the ***m*.coL** value to the second row,...and continue in this sequence. (The sequence does not influence the final result but the method reduces the probability of making mistakes)	

PASS.	**Makes the ConF. page read-only. Holding down for 1 minute an EnAb ↔ dISA change occurs and the ConF. page will be invisible and after repeating it visible.** **Setting range: 0.0.0.1. - 9.9.9.9.**

10. Flowcharts

You can find the most complex flowcharts on the next pages. There is not a general method for configuration sequence. We propose, make copies from the important pages of this manual and see together the information without turning pages.



7	6	5	4	3	2	1	0	ConF/Cnt*/dEF where * = 1, 2, 3 and 4
						0	0	There are not relays and SSd outputs
						0	1	Control with **one** relay or SSd
						1	0	HEAT-COOL control.
						1	1	Motorized valve positioner
					0			Heating (inverse control)
					1			Cooling (direct control)
				1				Enabling the max value **Stnd./Y*** actuator when **Stnd./ALrb/AlbL[¤]=1**, where *=1...4 and ¤=0...3, equally or cross-connected motor driver, when using it (xxxxxx11).
	0	0	0					The input of * Control Block is **In 1**
	0	0	1					The input of * Control Block is **In 2**
	0	1	0					The input of * Control Block is **In 3**
	0	1	1					The input of * Control Block is **In 4**
	1	0	0					The input of * Control Block is **In 5**
	1	0	1					The input of * Control Block is **In 6**
	1	1	0					The input of * Control Block is **In'7**
	1	1	1					The input of * Control Block is **In'8**
1								The * Control Block is always **ON**, where * = 1, 2, 3 and 4

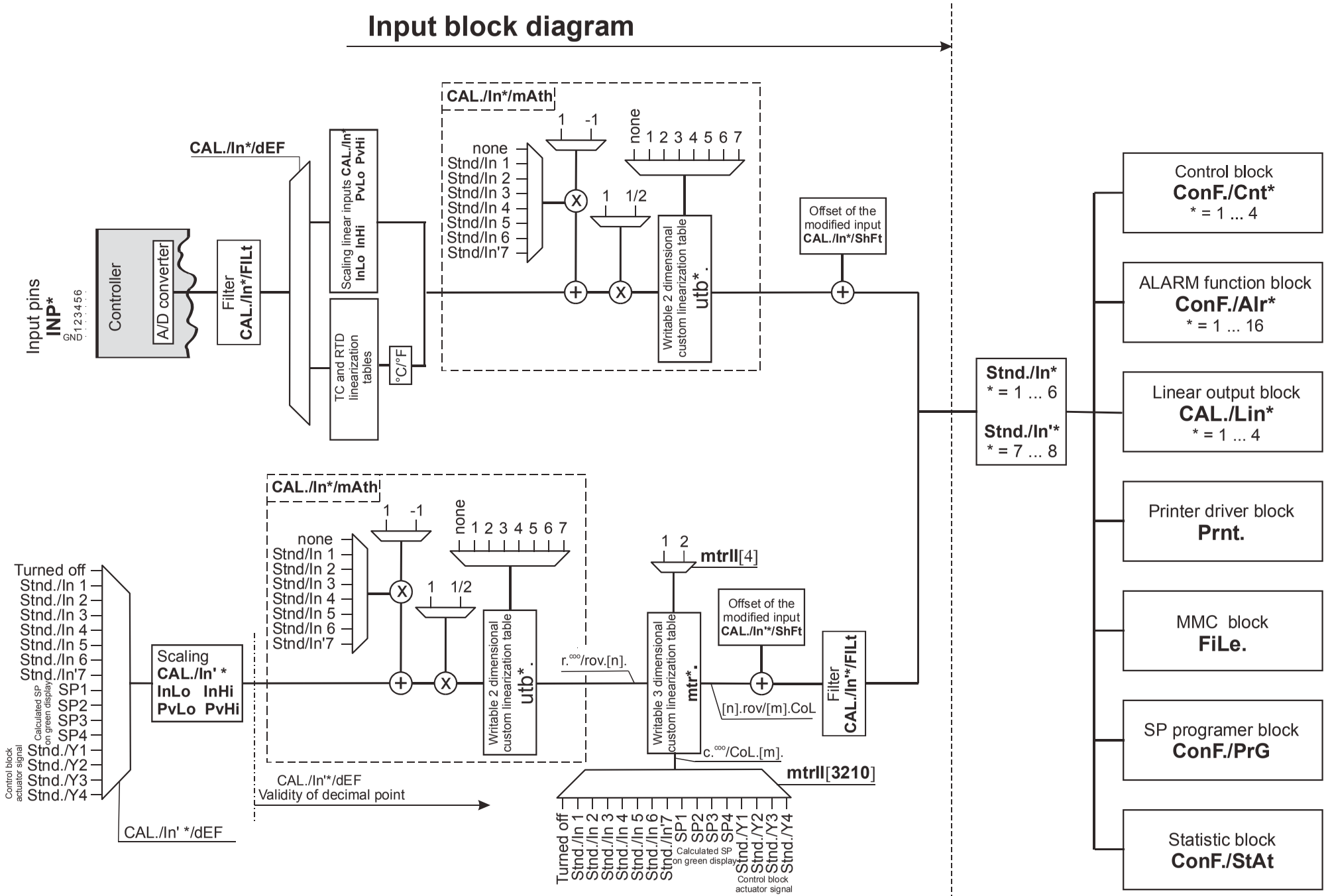
where * = 1, 2, 3 and 4

7	6	5	4	3	2	1	0	ConF/Cnt*/SEt where * = 1, 2, 3 and 4
						0	0	SP source selection **SP+0** normal, without adding
						0	1	SP source selection **SP+In 6** for using remote SP, cascade
						1	0	SP source selection **SP+In'7** for using math relations, etc
						1	1	SP source selection **SP+In'8** for using math relations, etc
					1			SP chosen from digital inputs
				1				Adding the computed value of **SP1** to the **SP** of the * Control Block (**PAr*./SP**), where * = 2, 3 and 4
			0	0				There is 1 PID set only
			0	1				There are 4 PID sets with outer selection
			1	0				There are 16 PID sets with selection by the actual SP (spaced equal among the configured **SPLo** and **SPHi**)
			1	1				There are 16 PID sets with selection by the **EvnH[7654]** event codes, where 0000=1...1111=G
	1							Every PID set has an own **Yt.** and **cYt**
1								PID sets are invisible

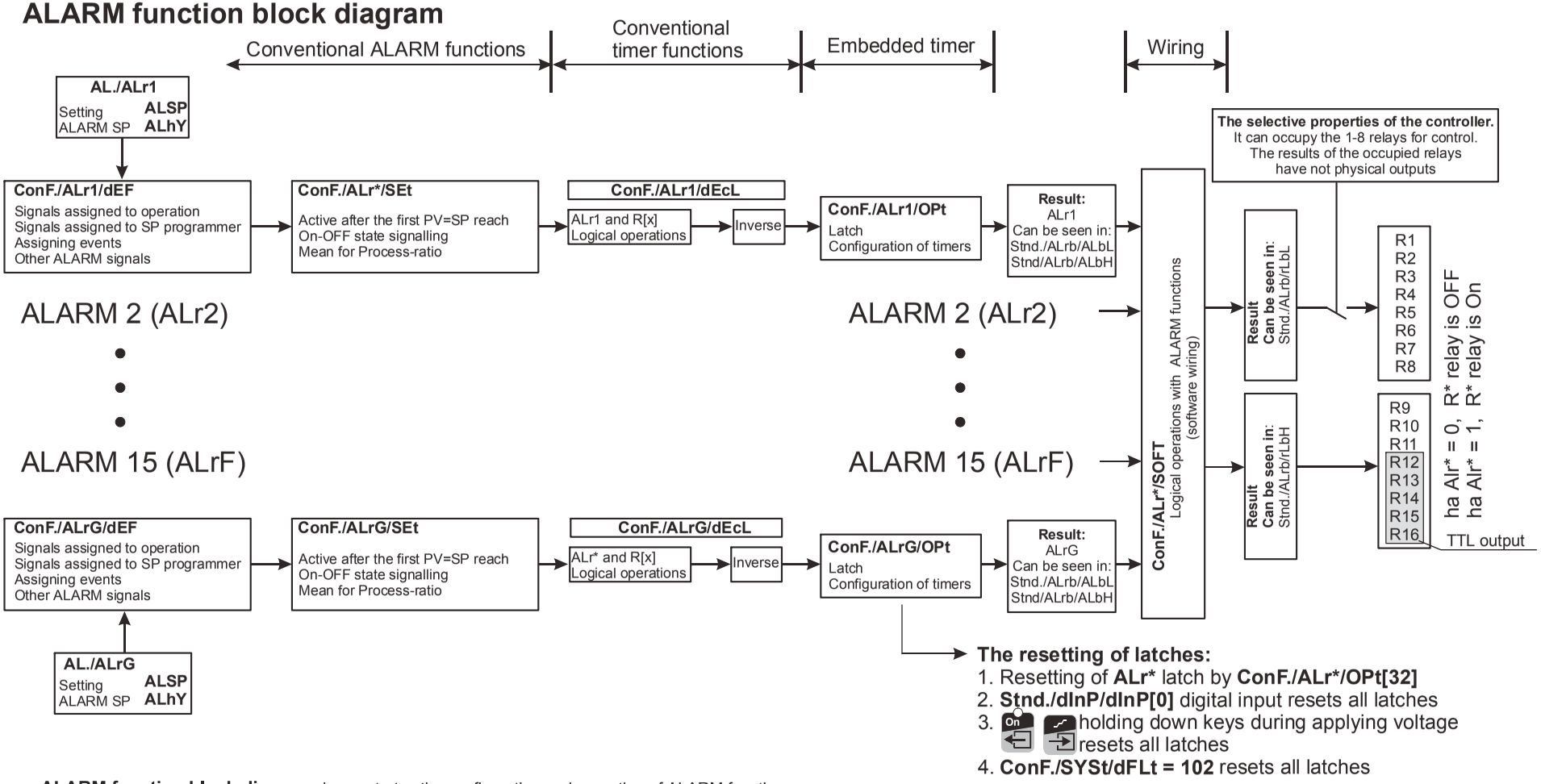
The SP programmer does not belong to the Control Block.
It can be configured at **ConF./PrG**

7 When the normal SP programmer operates (**Conf./Prg/dEF[10] = 10**) this switch is open for the 2, 3, 4, Control Blocks.
The offset of the 2, 3, 4, Control Blocks to the first can be set at **PAr*./SP** page where * = 2, 3,and 4

Input block diagram



ALARM function block diagram



ALARM function block diagram demonstrates the configuration and operation of ALARM functions.
The inversion of an ALARM is valid for the settings by the dEF, the SEt and the dEcL tables. The next settings hereafter use these results.
The result of the ALARM (ALr*) function may be 0, or 1. The result can be used in two modes. If we do not apply logical operators, than this value automatically will be connected the appropriate relay, so ALr* = R*.
The relays working as control outputs can not be ALARM outputs. We can refer them but they can not appear as outputs. The other using mode is the connection by BOOL rules. The result comes from the BOOL calculation.

Summarized, the control outputs (control blocks) force the system to work properly. We adjust the these by configuration. The control blocks have an overall priority. They occupy automatically their outputs.
E.g. the first control block occupies the R1 and R4 relays, when a motor actuator was set. In case these relays can be used in logical connections only.

11. Using SP programmer

The controller measures the error [e], which is the difference of set-point [SP] and process value [PV]. By the usage of the signs in KD9 controller the error is: e = SP - PV. The controller decreases this error by the calculated output signal, named: Y. There are two base working mode by the error type:

1. The SP is constant the error depends on PV only. After reaching steady state the error becomes zero. So when the system is on steady-state it will work with steady-state output signal and it will be only altered by disturbance. We name this mode: continuous process control (or steady state control). **The continuous process controller operates with constant SP.**
2. Many technologies need continuously changing SP. Annealing, chemical processes, sewage/waste-water cleaning, sowing-seed drying, mushroom cultivation, etc. are such technologies. In these systems the SP changes by the time so when they reach a steady-state an error occurs which is SP1-SP2. The SP is given by the **SP programmer.**

11.1. The SP programmer

An advanced SP programmer can fulfill lots of tasks beside the SP generation. The operations will be configured by the customer. You have purchased a universal compact controller, from you can "manufacture" a controller which will fit best for your technology. The "manufacture" consists of configuring, setting parameters and writing SP program. The SP programmer is a block which cooperates with the other blocks. We must mention here again that the selectivity simplifies the configuration because the invalid menu items do not appear. When using SP programmer configure it first please.

You can set the properties of the SP programmer in the **ConF./PrG/… Configuration Tables.** There are references for the settings.

The SP programmer has 4 working mode by the **ConF./ProG/dEF[10]**

1. **[00]** there is not an SP programmer. The controller is in continuous process mode. The menu items of the SP programmer do not appear.
2. **[01] sequencer.** The controller switches the relays by the program in function of time, independent of controlling the process. It is applicable to execute events in succession. You can program conditions, setpoints, waitings, cycles etc. in the process by the **FLAG** -s. E.g. One door opens by the sensor built in the floor, after it a switch turns on the light, after writing a code in a device the next door opens, after 2 minutes the light turns off and the program returns to the start position. During these operation the controller controls the temperature and humidity of the room.

The Figure 9 shows the operating scheme

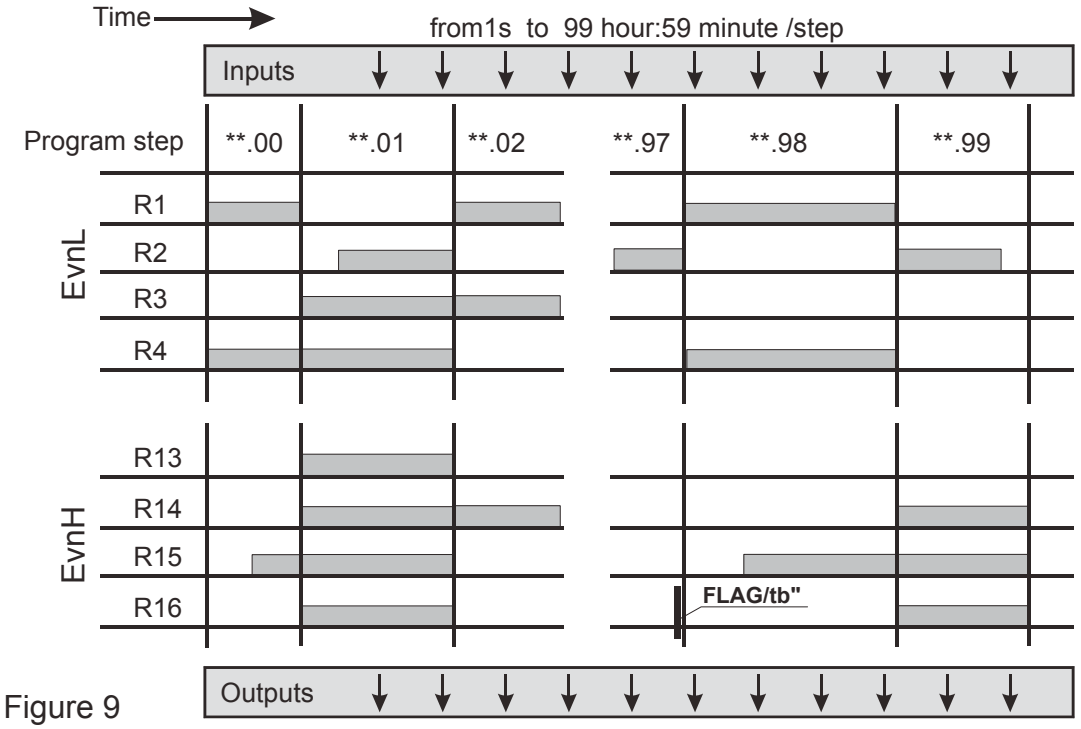


Figure 9

1. **[10] normal SP programmer.** The system is controlled by a written program. There are more possibilities to alter the SP. The program executes conditions, branches, cycles, etc. by the **FLAG** instructions. The program can only overwrite the SP of the 1. control block (1. channel). Using more channels (e.g. multi-zoned devices) in which the SP of every zones follows the first, all of the other SP of a control block can follow the 1. control block by the configuration. The SP of the other blocks (2., 3. and 4.) can be shifted to the SP of 1. control block in every program step (segment) each and all with varied values. (Can be set at PAr*/SP)

The working principle can be seen in Figure 10

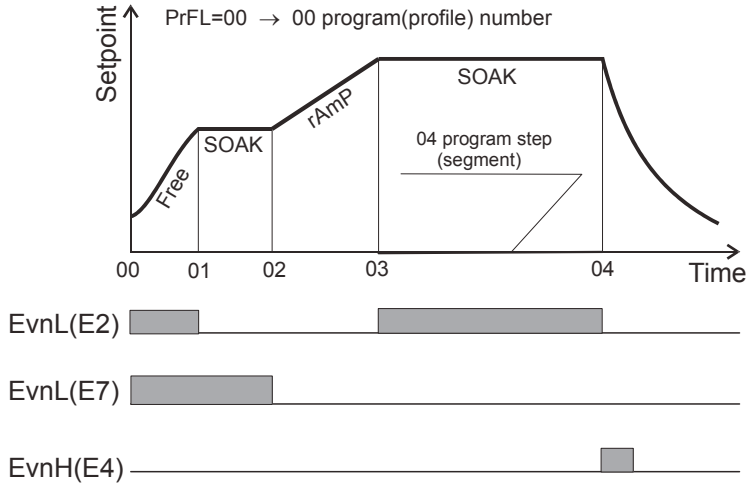


Figure 10

3. **[11] 4 SP programmer with the same time base.** The controller can control every valid control block by an separate SP programmer with same time base. So it is useful, when the SP has to change separately in every control block. Because of the same time base, the instructions are not equal with the ones of normal SP programmer.

The working principle can be seen in Figure 11

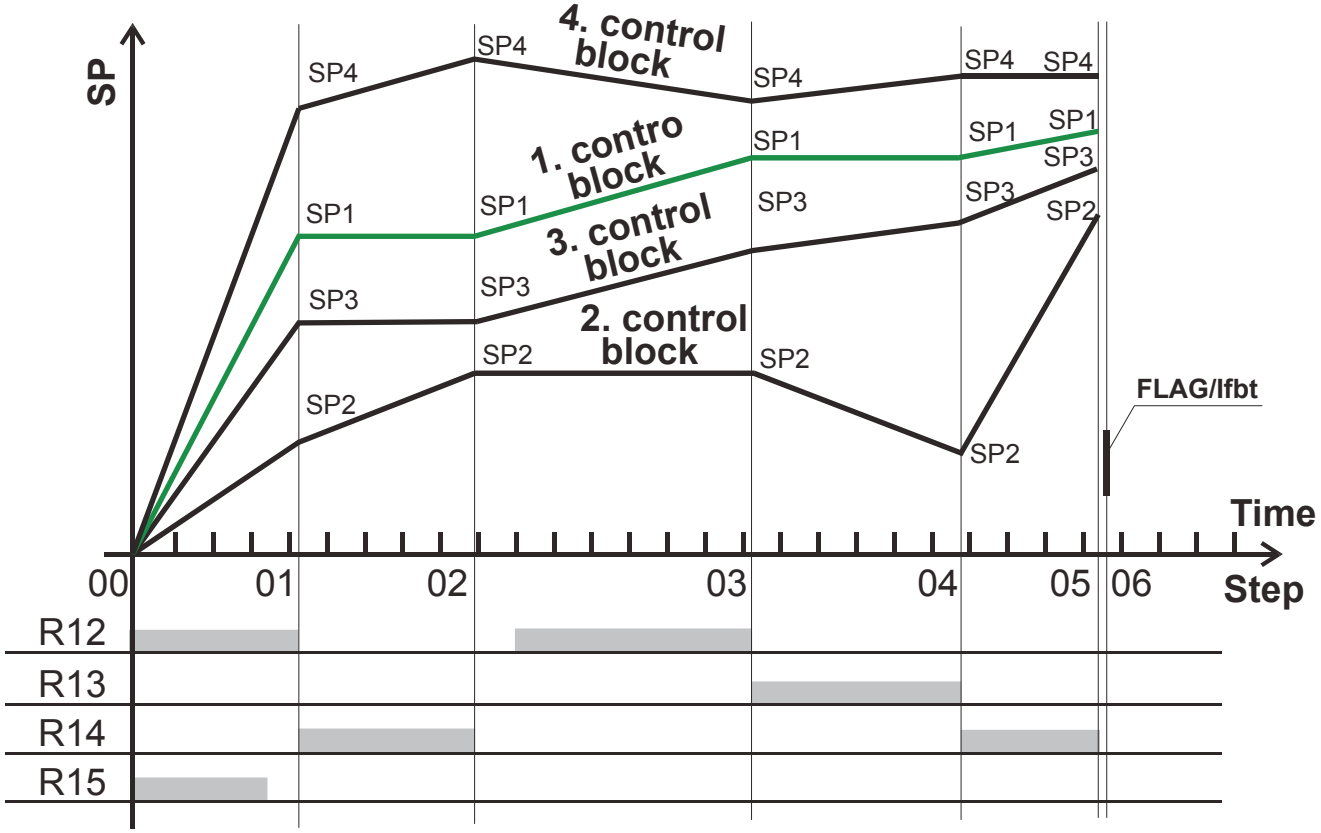


Figure 11

Writing a program(profile)

Enter with 2nd keys into the SP program writing. Everything will operate by the configuration as it was done. There may be that some instructions you need do not appear. If they are necessary verify the configuration. The SP writing is a very time-consuming process so it is advisable not to configure unneeded menu items. E.g. do not configure 4 SP programmer when you need a normal. If you configure it, the programmer will offer you 4 SP in every step, which are absolutely unneeded.

All SP program data can be seen on the display. This table shows the menu items for program writing:

ConF./PrG/dEF[10] =			
00 none	**01** sequencer	**10** normal SP programmer	**11** 4 SP programmer
-	**timE**	**timE**	**timE**
-	-	**rAmP**	-
-	-	**SOAK**	-
-	**FLAG**	**FLAG**	**FLAG**
-	-	**SP1**	**SP1**
-	-	-	**SP2**
-	-	-	**SP3**
-	-	-	**SP4**
-	**EvnL**	**EvnL**	**EvnL**
-	**EvnH**	**EvnH**	**EvnH**

Menu items appear in order as showed in the columns of table. The SP programmer handles all elements of the system. So the program writing is not enough to write the data of menu items contained the table above, but you must configure all parts of the system. There is not a defined sequence in configuration. E.g. you can configure an ALARM which is actuated by an event before or after the SP program writing. But the menu is selective here too. These menu items do not appear which were not configured, or the options which were not purchased.

The menu items of SP programming: (time base for the time axis in: ConF./PrG/OPt[10])

timE is one of the **SP** change possibilities. We can give the changing time and the SP at the end of the segment. The controller calculates the SP change rate and changes the SP by this "velocity". If you enabled the setting in running mode the SP programmer recalculates these data by the new values. The action of **timE** if you enabled the setting in running mode the SP programmer recalculates these data by the new values.

Signal of **timE** on the display:
e.g.: 05:14
(without decimal point)

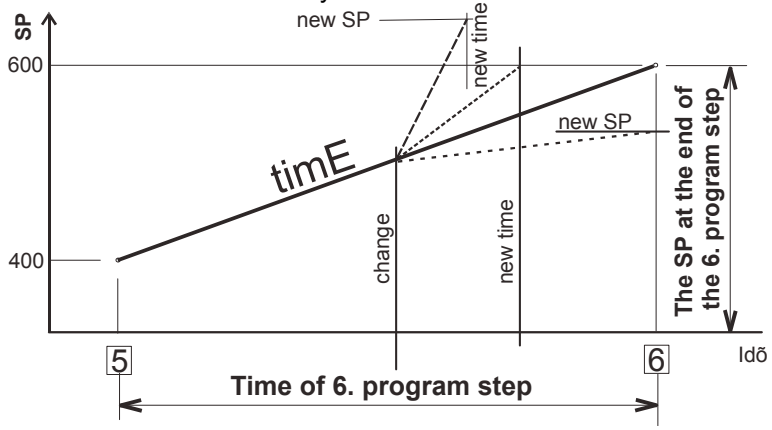


Figure 12

rAmP is another possibility of **SP** changing. The SP changes by a given "velocity" to a given value. If you enabled the setting in running mode the SP programmer recalculates these data by the new values. The action of **rAmP** if you enabled the setting in running mode the SP programmer recalculates these data by the new values.:

Signal of **rAmP** on the display:
e.g.: 060.0
(it is not a decimal point!)

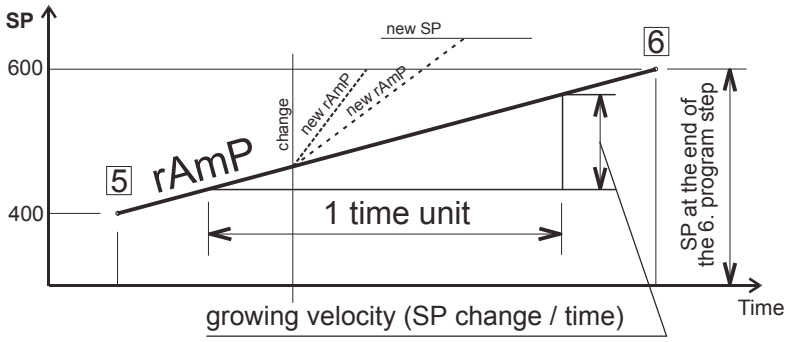


Figure 13

SOAK When the program steps into the (n)th segment than its SP jumps to the SP1 value. The PV reaches the SP1, the timing begins and its duration is the SOAK value. After this timing comes the (n+1)th segment. The action of SOAK if you enabled the setting in running mode, the SP programmer recalculates these data by the new values. The action of SOAK an be seen in Figure 14:

Signal of **SOAK on the display**:
pl.: 00:00.
(it is not a decimal point!)

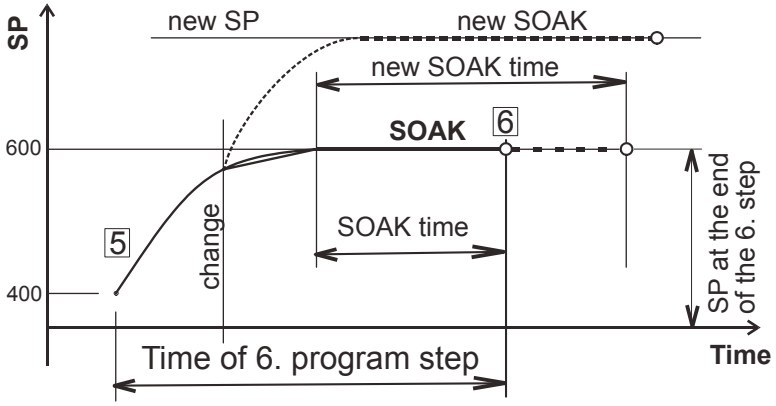


Figure 14

11.2. FLAG

FLAG is not an instruction, but a group of "HAGA-BASIC commands". You can call the FLAG group in which you find the instructions. See them in the table below:

FLAG	Descriptions	Value
noP	Empty instruction. Replaces unnecessary (deleted) SP program segments	-
FrE*	Overwrites the SP of *control block during the segment with the given value Where: * = 1, 2, 3, 4, The SP programmer or the sequencer waits for the PV=(green)SP state	**PAr*./SP** = ****
End	The SP programmer closes the program and exits.	-
Goto	The SP programmer jumps to the given address	**.**
CALL	Subroutine call. The SP programmer jumps to the given address. After execution the program returns to the next segment (after the CALL instruction). Only 1 stack depth.	**.**
rEtn	The last instruction in subroutine. It points to the segment after the **CALL** instruction.	-
SP*	Rewrites the SP of 1, 2, 3, 4 control block during a segment with the given value. Only in normal SP programmer or sequencer mode.	**PAr*./SP** = ****
tb"	The time base from this segment is: **minute:second**	-
tb'	The time base from this segment is **hour : minute**	-
tb-h	The time base from this segment is **day : hour**	-
Sto*	Inputs the given value into the *-th register, where * = 1, 2, 3, 4	****
dEc*	Decrements the *-th register by 1, where * = 1, 2, 3, 4	-
IFc*	If the value of the *-th register equals with the given value jumps over the next segment. (Jumps from (n)th to (n+2)th segment)	****
IFi	If the assigned (by EDS) digital inputs have a value: 1, jumps over the next segment. Can be seen in: **Stnd./dInp**	
IFAL	If the value of the ALr* is 1, jumps over the next segment. Where: * = 1 … 8 Can be seen in: **Stnd./ALbL**	
IFAH	If the value of the ALr* is 1, jumps over the next segment. Where * = 9 … G Can be seen in: **Stnd./ALbH**	
IFrL	If the value of the R* (relay) is 1, jumps over the next segment. Where * = 1 … 8 Can be seen in: **Stnd./rLbL**	
IFrH	If the value of the R* (relay) is 1, jumps over the next segment. Where * = 9 … 16 Can be seen in: **Stnd./rLbH**	
IFb1	When the condition is right jumps 1, when false than jumps 2 by pushing keys together.	-
IFb2	When the condition is right jumps 1, when false than jumps 2 by pushing keys together.	-
St_n	Starts the autotune process in the assigned (EDS) control block.	4 3 2 1
IFtn	Monitors the autotune process in the assigned (EDS) control block. When it is finished jumps over the next segment.	4 3 2 1
mmrS	Copies its content (EDS) in **StAt./rSEt**	4 3 2 1 4 3 2 1
rCtr	Clears the content of **Stnd./dInP/Ctr*** to zero [00000000]	8 7 6 5 4 3 2 1
ASP*	Where*=1…8. rewrites the value of **ConF./ALr*/ALSP**. The **ConF./Alr*/dEcL[0]=1** enables the **FLAG**	**ConF./ALr*/ALSP**
AtL*	Where*=1…8. rewrites the value of **ConF./ALr*/ALdt**. The **ConF./Alr*/dEcL[1]=1** enables the **FLAG**	**ConF./ALr*/ALdt**
AtH*	Where*=1…8. rewrites the value of **ConF./ALr*/AHdt**. The **ConF./Alr*/dEcL[1]=1** enables the **FLAG**	**ConF./ALr*/AHdt**
SSP*	Where*=1…4. rewrites the value of **ConF./PrG/SSP*** with the written one in **FLAG**.	**ConF./PrG/SSP***
SnoP	Synchronized empty instruction. Occupies the places of cleared instructions.	-
WtrL **WtrH**	Conform to **IFrL** and **IFrH FLAG-s** with the difference that in the EDS assigned relay state is 0 will wait in this program step. When its value changes to 1 continues the program.	
--.Go	It goes in the running program to the xx assigned progam step. ##.xx, where ## is the program number and xx the step number. The ## has not any function.	##.xx

The conditional checking of EDS values may be logically **AND** **OR**

The synchronized instructions will be executed by seconds. The controller executes the unsynchronized instructions max. five times in a second. The **Snop** and **FrE*** instructions are always synchronized. The synchronous properties can be set at **ConF./PrG/SET[2]**

11.3. The events

The advanced SP programmer can execute events during running. These events are valid in a segment which they are set in. E.g. the SP programmer in segment 03 activates the R7 and R8 relays and inactivates the R16. The events can be assigned to any ALARM function. (see figure 15). The ALARM-s have uncountable variations and they can be linked by Boolean relations which give exceptional abilities for the SP programmer.

The SP programmer may have separate event code set for every profile. But all can use the event code set of the 00 profile.

You can choose 16 sort of events for every segment. One profile has 100 segment so a process can be divided into 100 parts. E.g. you can fill a tank to predetermined levels with different liquids and mix them, while the pressure and temperature are controlled by the time. This process is a batch-control.

Figure 15 shows the set of events. We propose the sequence of setting by this figure:

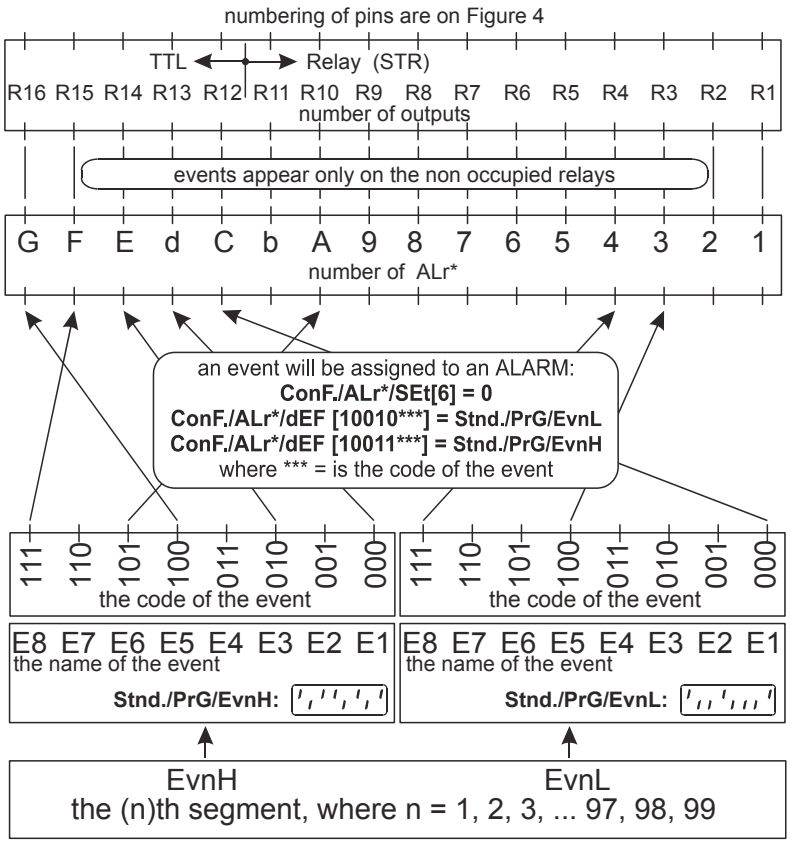

numbering of pins are on Figure 4
TTL
Relay (STR)
R16 R15 R14 R13 R12 R11 R10 R9 R8 R7 R6 R5 R4 R3 R2 R1
number of outputs
events appear only on the non occupied relays
G F E d C b A 9 8 7 6 5 4 3 2 1
number of ALr*
an event will be assigned to an ALARM:
ConF./ALr*/SEt[6] = 0
ConF./ALr*/dEF [10010***] = Stnd./PrG/EvnL
ConF./ALr*/dEF [10011***] = Stnd./PrG/EvnH
where *** = is the code of the event
111 110 101 100 011 010 001 000
the code of the event
111 110 101 100 011 010 001 000
the code of the event
E8 E7 E6 E5 E4 E3 E2 E1
the name of the event
Stnd./PrG/EvnH:
E8 E7 E6 E5 E4 E3 E2 E1
the name of the event
Stnd./PrG/EvnL:
EvnH
EvnL
the (n)th segment, where n = 1, 2, 3, ... 97, 98, 99

Figure 15

1. Configure the ALARM functions by the section of **Configuration tables** in **ConF./ALr*/…** . You ought to plan and write down the ALARM functions in complicated big systems. Use the **ALARM function diagram**. There can be seen that after setting **Conf./ALr8/SEt[6] = 0** and **ConF./ALr*/dEF[1001****]** a lot of properties can be chosen. The section of **The ALARM action** deals with them in details.

   E.g. the ALr7 activates the R7 relay and opens the HG valve with 20 s delay
   the ALr8 deactivates the R8 relay, if ALrA and ALrG are active than due this opens the V11 valve
   ALr6, ALr5, etc.

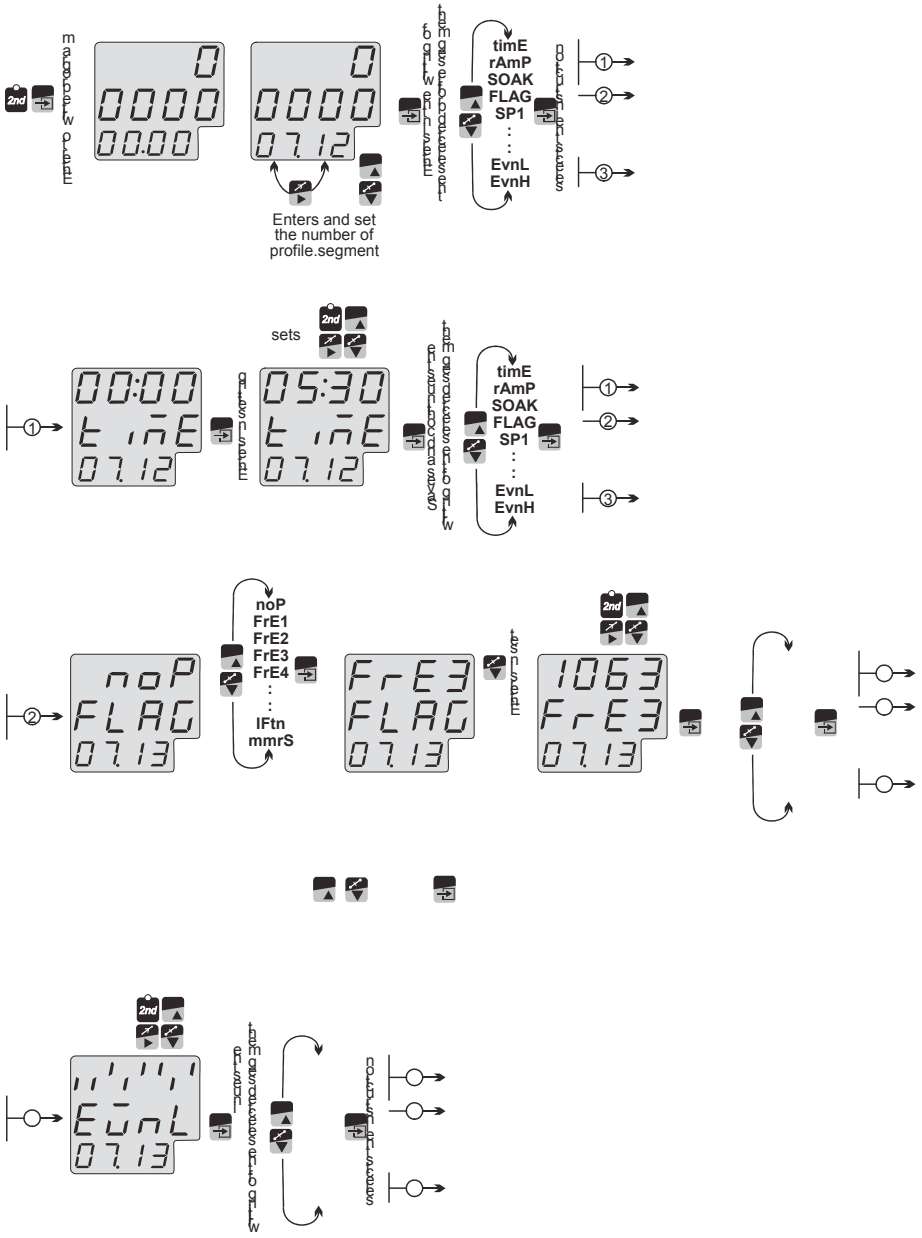
2. The table below summarizes the events **EvnH**(E8 …E1) and the **ALr*** ALARM-s. You can assign any **ALr*** to every event. But strongly proposed to choose the same index for both. You could overlook it more easily. (In this table we mixed the indexes to show the big abilities.)

Stnd./PrG/EvnH	E8 [7]	E7 [6]	E6 [5]	E5 [4]	E4 [3]	E3 [2]	E2 [1]	E1 [0]
EvnH code	111	110	101	100	011	010	001	000
ALr*	**ALrG**	**ALrF**	**ALrE**	**ALrd**	**ALrC**	**ALrb**	**ALrA**	**ALr9**

Associate the events with the ALARM-s by the table. (You can mix the lower row anyway). Set **Conf./ALr8/SEt[6] = 0**. This is the default value. After it set **Conf./ALr8/dEF = 10011111**. By these settings the **EvnH/E8** event activates the **R8** relay through **ALr8** if it is not engaged.

3. Set in one segment (e.g. **07.13** segment) the bit of **EvnH(E8)** to 1, that is pull up the bit in the EDS . Hereafter when the **07.13** segment runs the **ALr8** will be active (its value equals 1) and if there is a free **R8** it will be active too. The relay will turn on if it is in normal mode, or turns off if it is in reverse mode.

11.4. Course of writing SP program



Example
Profile number: 03

Segment number	timE	rAmP	SOAK	FLAG instruction	FLAG value	SP1	EvnL	EvnH
00				Sto1	4		00000001	10000000
01				FrE1	100		00000001	10000000
02		20.0				300	00000001	01000000
03			01:30			400	10001000	00000000
04	02:30					600	00000001	10000000
05				dEc1			00001000	10000000
06				IFc1	0		00001000	10000000
07				Goto	03.01		00001000	10000000
08				End			00001000	10000000

This program changes the SP by Figure 16. It repeats the profile three times. The relays are turning on and off by the event codes.

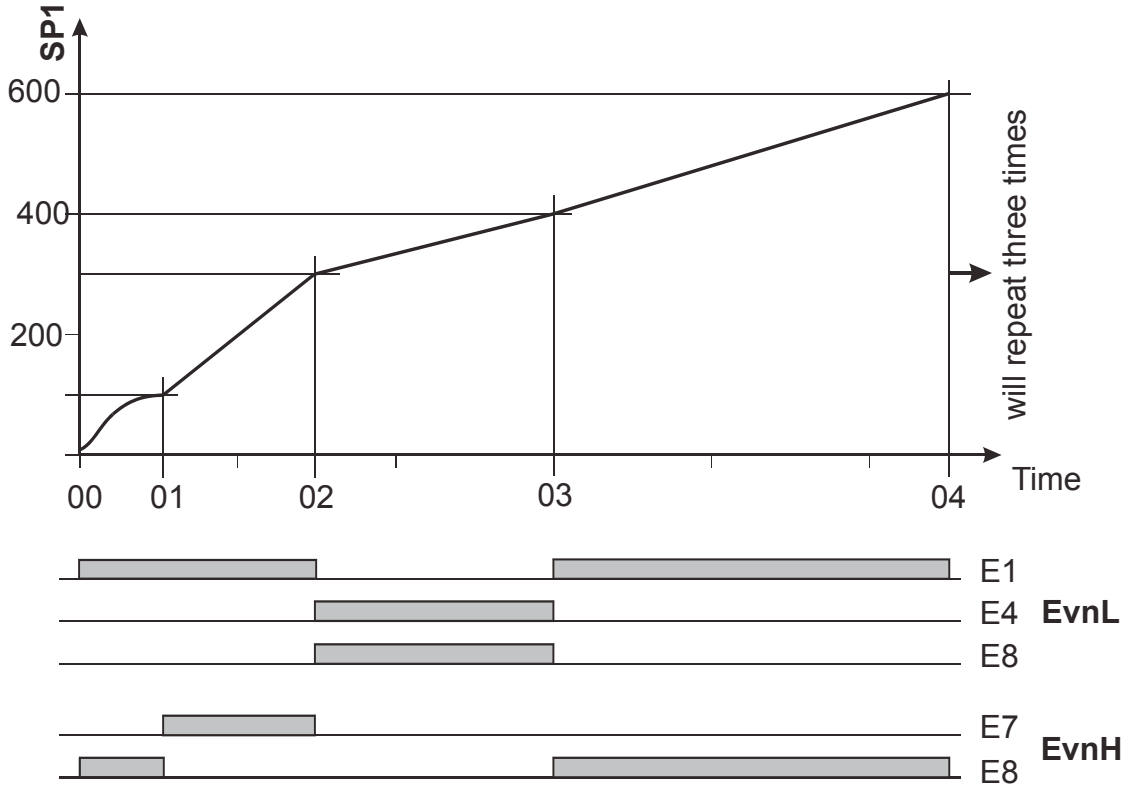


Figure 16

11.5. Special functions:

11.5.1. AUTO-WAIT

The AUTO-WAIT function is a restoring part of the SP program. At power fail the state of the system is out of order. Decreases, or increases the temperature, pressure, etc. After the power recovers the system values do not equal the SP programmer values. If **ConF./PrG/SEt[5] = 0** the programmer waits until PV=SP equation will be valid.

11.5.2. SHADOW

The SHADOW function sets some actuations in controllers which are connected together with communication line. In every slave (**ConF./Com1/dEF[6] = 1**) can be set the SHADOW function in **ConF./PrG/OPt[3] = 1**. After setting the slave will operate in two modes as written below:

1. The MASTER sends its calculated **SP1**, the **on/OFF**, and the **profile.segment** values to the SLAVES-s. If the SLAVE contains the same program as the MASTER, than it uses only its event codes and the others not. So the SLAVE operates like the MASTER, but may execute 16 own events. You need not to write the MASTER SP program in the SLAVE, it is enough only to write the event codes in.
2. You must write precisely the MASTER SP program in the SLAVE. After configuring the **ConF./Com1/SEt[3] = 1** the SLAVE will use its own events but it is ready to continue controlling stand-alone when the communication breaks.

The MASTER can fail not only on account of a communication break. The SHADOW mode can be monitored by so configured ALARM: **ConF./ALr*/SEt[6] = 1** and **ConF./ALr*/dEF = 10000000**. You can do with this ALARM any change-over or e.g. to turn off the MASTER.

You can use **hot swap** function by SHADOW mode with two KD9 controller with the same configuration and SP programs in very important or dangerous systems. You must wire and commission the controllers so that all inputs and outputs act accordingly to the MASTER after change-over. All of the controllers must have different communication addresses

11.5.3. The ALARM action (configuring in Conf./ALr*)

The ALARM is a Boolean function. The result of function by the fulfilled condition may be 0 or 1. There are 16 ALARM-s in the KD9, numbered 1, 2…9, A, b,…G. So the signal of the 12th ALARM is **ALrC**.

The ALARM input is that which you have configured. E.g. the **ALrC = 00000011** configured ALARM monitors the system errors and if such error occurs its output will be **1**.

An ALARM may assigned to relay, SSd, or TTL output. You can select ALARM by **CAL./Lin*/dEF[210]** bits, which gives 0 or 20 mA current on linear output and may actuate an SSR. The mentioned physical output can be found on the rear of the KD9 and their numbers equal with their decimal numbers (R1, R2…R16). You can connect the ALARM-s by Boolean rules. Setting R5 = ALr5 AND (ALr6 OR ALr7), if ALr5 = 1 and either (AlLr6 or ALr7) is 1 the output result is 1.

The ALARM is not a relay! It has not any physical output! The ALARM-s **may actuate** relays if they are assigned to them.

The mathematical definition: the relays actuate by the truth table valid for the ALARM-s.

The description of ALARM function

Input		Monitors	Output	Result (R* state)
Conf./ALr*/dEF [********] Conf./ALr*/SEt [********] Conf./ALr*/dEcL [********] Conf./Alr*/OPt [********] Conf./ALr*/SOFt [********]	configured	unfulfilled	=0	output inactive
		fulfilled	=1	output active

Note: The relays may be configured to normal or inverse action. The reverse relays are open in active state.

The controlled system changes continually. The controller forces the system to operate by the configured mode to have it work properly. Therefore, during the working time many changes will occur. These changes are called events. E.g., such a change may be the heat up to 250 ºC of a point in a system. If the **ALr3/dEF=[00100010]** and the **AL/ALr3/ALSP=250** outputs will be 1, than the ALr3 output will be 1 too and if the R3 in not engaged will be active. So you can see, that the controller makes different changes by an event on the system.

11.5.4. Settings of ALARM-s

Settings of ALARM-s are divided in four groups, which can be found in the Configuration tables.

11.5.5. System events, common events referring to the control:

- monitoring the communication (event = there is communication)
- monitoring autotune and hand mode and etc.
- monitoring input and system errors
- monitoring digital inputs and all of the ALARM states and signaling and interrogation

Common events referring to the SP programmer

- monitoring the different acting modes of SP programmer (Hold, run, autowait…..)
- monitoring the running events (rAmP, SOAK, FLAG……)
- monitoring the start delay period

Events defined in program segments

Every segment contains an event code EDS with 16 switches. You can control with these different logical events (PLC) through the proper output, while the SP program is running.

Events for the common input and output functions

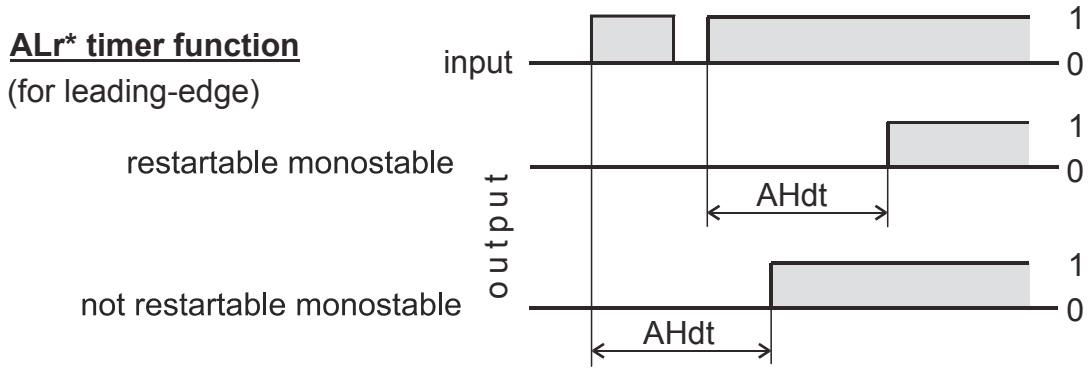
- **Process** turns on when its value reaches the **In***, **SP** and **Y** values
- **Processratio** turns on when changing rate reaches the **In***, **SP** and **Y** changing rate values
- **Deviation** turns on when deviation is bigger or smaller than the given value by the **In***
- **Band** turns on when band is bigger or smaller than the given value by the **In***

You can give signals trough the activated outputs for outer devices or for inner blocks by an selected ALARM. The output can be modified by the on or off state of the controller and the normal or inverse state of the relay.

12. ALARM timer function

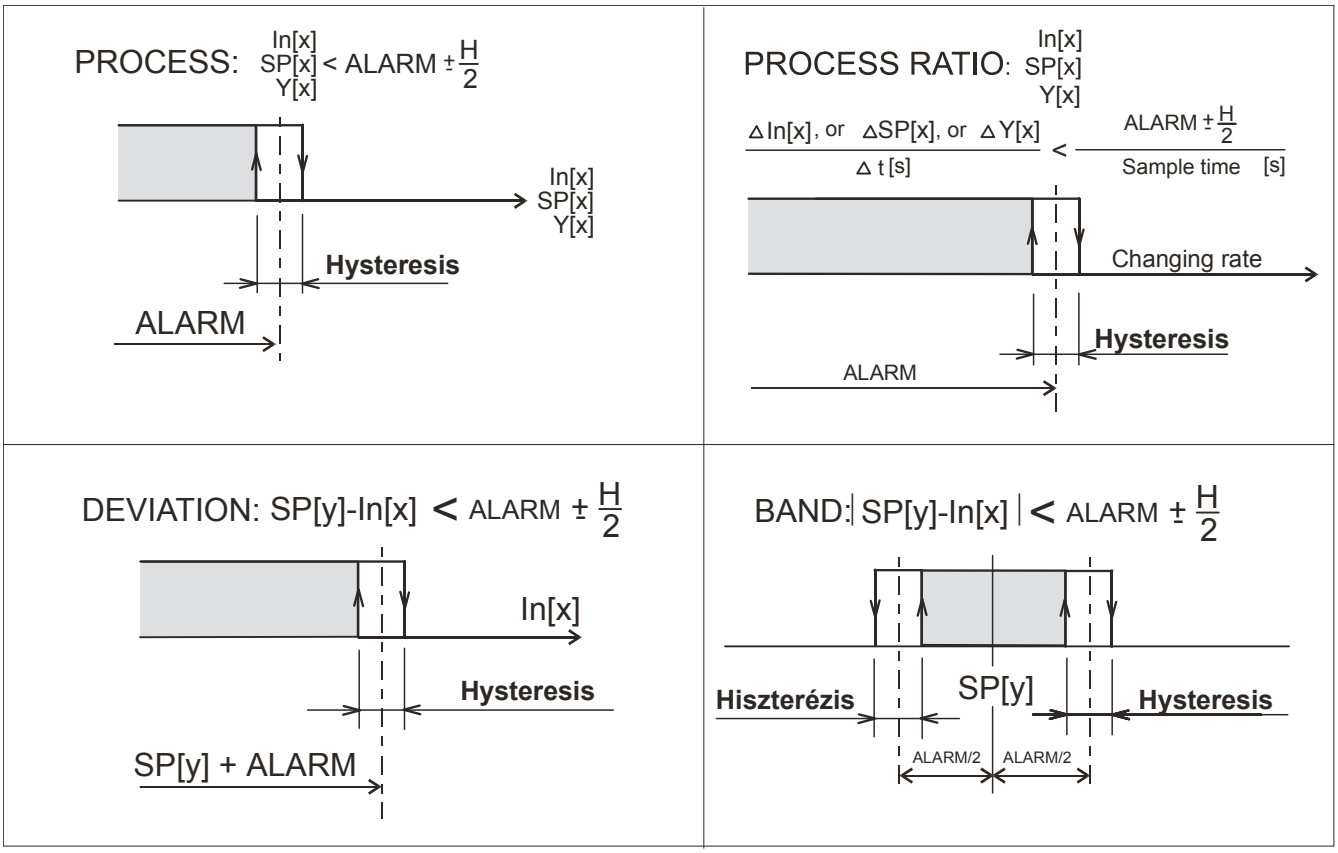
Every ALARM has a timer with latch so the action can be influenced in time. E.g. after a change 0 → 1, or 1 → 0 the action can be delayed by the rules of timer. You can use a latch, which holds the ALARM in activated state until its reset independently of its original configuration. There is more possibility to reset a latch.

Acting of a latch can be seen in Figure 17.



17. ábra

You can chose symmetrical and asymmetrical hysteresis for an ALARM in **Conf./ALr*/dEF[7]**. The acting of hysteresis can be seen in Figure 18.



The values of the ALARM and its hysteresis can be given in **AL.**

Figure 18

The acting principle of the hystreresis can be seen in Figure 19.

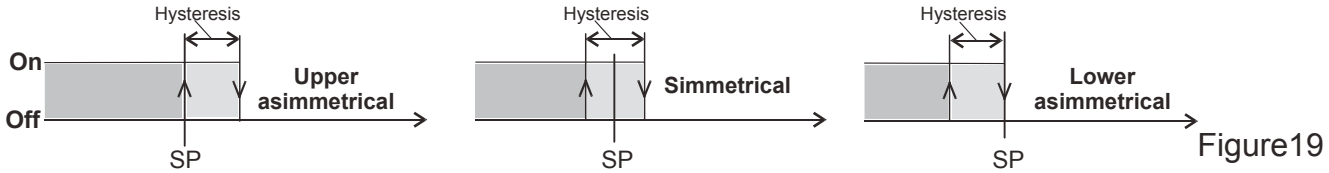


Figure19

It is strongly proposed to configure ALARM-s by **"ALARM function block diagram"**. You can see the detailed data in **Conf./ALr*** tables.

13. PID control

The four control blocks can be tuned independently as PID control channels (**Cnt1**, **Cnt2**, **Cnt3**, and **Cnt4**). Because of the big variety of system properties the tune process parameters must be configured by the system. The pages which refer to tuning are written down below, can be find in **Figure 5** and in the "**Configuration tables**":

PAr*/Pid*
ConF./Cnt*/SEt[7654]
ConF./Cnt*/dEcL[5]
ConF./Cnt*/OPt[3]
ConF./Cnt*/Yt
ConF./ALr*/dEF=000010xx (if **ConF./ALr*/SEt[6] = 0**)
FLAG-s in SP programmer: **St-n IFtn**

Every channel can use one or more PID parameter set (proportional, derivate, integral and cycle time parameters). There is an optimal PID parameter set for every controlled system. It does not depend only of physical properties of the system but the technology influences it too. You ought to use different PID parameter set for melting and annealing metals. For melting is better to input much energy in the system without taking account the waving temperature, while annealing demands a precise temperature program.

So as you can see above the tuning process can be done by simple written rules. The autotune helps you to find the PID parameters, but after it you need fine tune the system. When you have a special task e.g. crystal growing, zone melting, laboratorial measurement, etc. you must be experienced or call a skilled expert.

13.1. The PID tuning

You can choose more type of process for tuning.

13.2. Manual tuning.

You can always rewrite the PID parameters in **PAr*./Pid***. The PWM cycle time (Yt.) can be set in **PAr*./Pid*/Yt.** or in **ConF./Cnt*/Yt.** or depending on the state of **ConF./Cnt*/SEt[6]**. The first channel has a factory set default parameter set. This is good for the control of a not too fast and not too slow furnace.

The manual tuning is the fastest method, but the fine-tuning requires much time. It is essential to use a recorder for this process. Before starting the tune check the configuration.

13.3. Autotuning

The controller has a very good autotune algorithm. The algorithm opens the control loop and makes three wave in on-off mode. The computed values are able to control the system. The fine-tune is necessary only for systems with very big lag time.

The autotune process must be started by hand for every PID set separately or grouped together by program.

1. **For separately autotune** set the control loop and the SP (giving the approximate range in which the parameters will work well). Than hold down the 2nd untill **tunE** appears an begins blinking. After it a **T** will light on the bezel blinking in **OFF** state an lighting in **On** state. Turn On the controller (to On mode) for autotuning if it was OFF when you had started the process. After the waving the **T** goes out and the computed parameters will be written into the memory. The previous data will be overwritten. It is advisable to read out and archive the new parameters.

2. **Autotune using reduced power** (for given SP). The system may be damaged by autotune while the system is in overshot period. The algorithm can limit the overshot to the enabled critical value. You can see the configuration below whit which the problem can be solved. This limitation is good for any other parameters too. The example shows how to use the method for limitation the Y (actuator value).

ConF./SYSt/mmi2[0] = 1	the ALARM bits appear on **Stnd.** page (for checking)
ConF./Cnt*/Yd' = Y_{max} %	upper limit of actuator (is valid between 25 … 100%)
ConF./Cnt*/dEF[3] = 1	enabling the maximazing
Stnd./ALrb/ALbL = 1	the EDS shows validity of maximazing

   The maximasing will be done by the state of the assigned ALARM. If the assigned ALARM = 1 the actuator (Y) may not be bigger than the value of **Yd'**. You can see the value of the assigned ALARM in **Stnd./ALrb/ALbL**. So the autotune will not use bigger **Y** than we enable for it.

   There are more setting possibilities. You can assign an ALARM by an event code which will maximize the actuator value (Y) in a program segment to the given value.

   If you want to maximize the actuator value (Y) in a range of control use the this configuration:

   ConF./ALr*/dEF[3] = 00100xxx assigns ALARM (Process In[x]) for giving operation limit

AL./ALr*/ALSP = value from which the maximazing is valid
ConF./ALr*/dEcL[7] = 0 → maximizes under value of ALSP
ConF./ALr*/dEcL[7] = 1 → maximizes above value of ALSP

3. The controller computes the parameters for **motorized valves** by the total travel time. It ignores the total travel time in case of Yt. < 10 s, but uses it if is bigger. The total travel time must be configured in **ConF./Cnt*/Yt.**, or **PAr*./Pid*/Yt.** by the motor technical data.

4. The grouped autotune is a complicated task. That systems which properties change during control need more PID sets by the SP. The heat technical systems are as well known. In such system all heattechnical properties change by the temperature. The lag time, the heat conduction of insulation, the heat capacity etc. influence the PID values. So it is necessary to use more PID sets.

   The KD9 controller may operate with 16 PID sets for the 4 channel each. This involves that a 4 zoned program controlled furnace may need 4x16 PID sets. For a very precise control that is the **big task** to tune the system. This task is almost impossible without mistakes, not to mention the time needs and boring work.

   Due to the up-to-date instruction set of the KD9, it can automate the grouped autotune process. The next program computes 16 PID sets.

Base settings :	**SPLo = 0**	**SPHi = 1600**	range of tune
	ConF./Cnt1/dEF[0] = 1		control with one relay
	ConF./Cnt1/SEt = 01100000		16 PID sets for selection by EVENT code

This program will make the autotune in process **Cnt1** between 0-1600 in 16 equal ranges at the set value e.g.: 80, 180, 280, 380, ... etc.

	00.00	**FrE1**	**80**
	00.01	**CALL**	**00.50**
	00.02	**FrE1**	**180**
	00.03	**CALL**	**00.50**
	00.04	**FrE1**	**280**
	00.05	**CALL**	**00.50**
	00.06	**FrE1**	**380**
	00.07	**CALL**	**00.50**
	00.08	**FrE1**	**480**
	00.09	**CALL**	**00.50**
	00.10	**FrE1**	**580**
	00.11	**CALL**	**00.50**
	00.12	**FrE1**	**680**
	00.13	**CALL**	**00.50**
	:		
	:		
	00.30	**FrE1**	**1580**
	00.31	**End**	
	:		
	:		
subroutine	**00.50**	**St-n**	**00000001**
	00.51	**IFtn**	**00000001**
	00.52	**Goto**	**00.51**
	00.53	**rtn**	

The program tunes the system and saves the computed PID parameters in **PAr*./Pid*** page. There are two possibilities to use the PID sets. They may be chosen by the actual SP, or by EVENT codes.

There are four PID channels in the KD9. They may control different systems, so it may be necessary to tune them each independently. You can use this tuning program, but do not forget to rewrite the base settings.

14. The digital inputs

The KD9 holds connection between itself and another outer device. The digital inputs are able to sense outer potential free contacts. If the combinations of these contacts exist, than the controller executes an instruction or changes operating mode. There are seven digital inputs. The configuration determines the influence of inputs. Take it account that select only one task for a contact. If you select two, both selected task will be executed.

There are counters for digital inputs activities configured in **CAL./dCtr/Ctr**.

7	6	5	4	3	2	1	0	**Digital inputs Di8** = [**7**] ... **Di**1 = [**0**] **Stnd./dInP/dInP**
x	x	x		—				Selection one of the first 8 program, where: x = 000 → 00, 001 → 01 … 111 → 07
			1	—				Stops the program "HOLD" (function of the key on bezel)
1				—				Sets to hand control in the 4. PID channel
	1			—				Sets to hand control in the 3. PID channel
		1		—				Sets to hand control in the 2. PID channel
			1	—				Sets to hand control in the 1. PID channel
				—	1			**On↔OFF** switch (function of the 2nd keys on bezel)
						y	y	**PAr*./m-SP/mSPy**, where: y = 00 → 1, 01 → 2, 10 → 3, 11 → 4 and * = 1, 2, 3, 4 (for every set channel)
				—			1	Resets all latches (by a push button or relay contact), if **ConF./SYSt/mmi2[6] = 1**

It can be seen in the table, that the [7654] and [1] bits are engaged in two operation, so it is strongly proposed to select only one.

The [**10**] bits set the same **mSP[y]** in each channel, where this selection function is enabled. (**ConF./Cnt*/SEt[2] = 1**)

The effect of [**765**] bits will be only valid if **ConF./PrG/SEt[32] = 11**.

The digital inputs can be checked in **Stnd*/dInP/dInP** page where the bits set by the table will appear if you have configured **ConF./SYSt/mmi2[21] = 11**. During the test, the controller does not use the switches connected to the rear pins.

It is proposed to use rotary code switch instead of xxx and yy contacts. The label on the switchboard informs you well about the valid state.

15. The digital outputs

You can configure 16 ALARM-s. The number of an ALARM equals with the number of its relay. Therefore, the ALARM 1 acts the relay 1, etc. There are maximum 11 relays in KD9 so the last 5 ALARM-s have not relays.

Because of lack of relays, the KD9 uses **digital outputs** for the last five ALARM-s, with TTL signal level.

16. Error, error messages, resets

The error

The controller must operate by the configuration and program. Any dysfunction causes an error state. The errors are grouped by their type. There are two groups, which have different effects.

You can configure these effects by configuration (stop running, warning, switch to another mode, use hot swap, etc). The configuring depends on the character of the system.

Error messages

The error message (mnemonic) appears on the display. The necessary actuating must be configured for this message. The message can be seen until the error exists and the action is valid too. You can disable the actuation of the error message of a control block by **ConF./Cnt*/dEcL[2] = 1**, but be careful it is a very dangerous state!

Error groups

Channel input error (error on the physical pins of sensor)

The KD9 sends an error message if a not configured value appears on the input. It may be an overflow or underflow. The cold junction error appears only as a warning. The channel (control block) error turns off only its channel, but you can configure only giving a warning without turning off. The channel (control block) error message contains the number of the channel. These error states can be queried in **Stnd./In*** and **In'***.

The input error of a channel will act as a system error if **CAL./ In*/dECL[5] = 1**. In this case, the KD9 turns off every channel.

The KD9 can turn on hand control the defective channel by configuration **ConF./Cnt*/dEcL**[**2**_0] = **1_1**. Reset the hand mode to control mode by pushing the 2nd key than the ▲ key for 5 s after the error have been repaired.

Hierarchy

Mnemonic		channel error
E.cAr	Warning. Cold junction error in a channel	Hierarchy grows ↓
Eund	Underflow in a channel. Input is smaller than the minimal value of the linearization table	
E.our	Overflow) in a channel. Input is bigger than the maximal value of the linearization table	

The error in the input block appears in the result. All the errors are inherited to the result. The error message, highest in hierarchy, will be seen on the display. E.g. if two errors occur in the same time, a cold junction error and a math channel overflow error, than the overflow error will appear.

System error

The damage of hardware or its malfunction can cause a system error. E.g., the memories have got imperfect data. The KD9 are always checking the running and when error occurs turns off the controller, puts it in off state. The error message appears on the green display. The error can be cleared off by a mains reset and turning on after it or clear by setting **Conf./SYSt/dFLt = 128**. If clearing was not successful, the hardware must be repaired.

Service: HAGA Automation Ltd 1037 Budapest, Királylaki út 35, Hungary www.hagamat.hu

Mnemonics on the display

Mnemonic		channel error
E.P.iL	Illegal program instruction. Repair the wrong program step (segment).	Hierarchy grows ↓
E.P.St	Program stack error. Repair the wrong program step (segment).	
E.Ain	Analogue input error	
E.E2P	Defective E2PROM	
E.bcP	Imperfect **Backed-up** data (which were saved at turning off). It can appear when turning on.	
E.CAL	The calibrated data are false, every measurement is inaccurate. Calibration may be done only in service.	
E.uCt	The load is to big in the processor.(**Stnd./LOAd**). Lower the load by rewrite some functions.	
E.-Ad	AD converter error	
E.rAn	RAM error inside the processor	
E.Pr0	Program error inside the processor (executive routine)	

The error messages can be processed in the controller. The settings of **ConF./ALr*/dEF** [210] activate an ALARM by the error. The ALARM-s can overcome the imperfect working due their properties and Boolean functions. You can get warnings or acting by relays.

The SP programmer uses the ALARM-s for warning and acting by the HAGA-Basic instructions and EVENT-s

17. Security

There are many data in the controller. They are very important for you. The system works by the configuration data and only one of them may cause a disaster. There are technological data too, which may be industrial secrets. Therefore it is strongly proposed you to hide your data inside the controller.

The software of the controller enables you to save your controller by disabling instructions an hiding configuration parameters. The setting has three levels:

invisible
visible, but read only
visible and writable

Write the password after configuration in **PASS. = X.X.X.X.** where **X** = **0 ... 9** and **0.0.0.0.** without password. The password hides the **ConF.** page and so you can reach the data only by the password. You can hide the **PASS.** page too.

WARNING! Do not forget the password. The forgotten password can be cleared by the manufacturer!

Function	Configure in:
Global inhibitions and authorizations	**ConF./SYSt/mmi***
There is not an SP programmer	**Conf./PrG/dEF[10] = 00**
Inhibitions for counters	**Conf./PrG/dEF[43]**
Inhibition of start delay	**Conf./PrG/dEF[5]**
Enabling HOLD function from digital input	**Conf./PrG/dEF[6]**
Inhibition of AUTOWAIT function	**Conf./PrG/SEt[5]**
Inhibition of NEXT button	**Conf./PrG/SEt[6]**
Inhibition of HOLD button	**Conf./PrG/SEt[7]**
Inhibition of program instructions	**Conf./PrG/dEcL**
PID parameter sets are invisible	**Conf./Cnt*/SEt[7]**
Inhibition of hand control, autotune, SP setting, visibility of control block data	**Conf./Cnt*/dEcL**
Enabling the statistics	**ConF./StAt[7654]**
This ALARM does not exist	**ConF./ALr*/dEF = 00000000**
The control block does not exist if	**SPLo ≥ SPHi**

18. Specific applications — Basic principles

The specific application examples below help understanding how this controller works.

The KD9 controller has a block structure. Every block is an independent unit and can be separately configured. You can configure the connections and coactions too. If the configuration conforms to the system, than the controller will work like a complete device inside with its well-configured blocks.

18.1. The blocks:

SP programmer
Configuration in: **Conf./PrG**

PID channels
Configuration in: **Conf./Cnt***

Statistic
Configuration in: **Conf./StAt**

ALARM
Configuration in: **Conf./ALr***

Communication
Configuration in: **Conf./Com1**

Input
Configuration in: **CAL./CJ** **CAL./In*** **CAL./In'***

Linear output
Configuration in: **CAL./Lin***

Printer
Configuration in: **Prnt.**

Data logger
Configuration in: **FiLE.**

The task assigns the practical sequence of configuration. E.g., so long you do not configure in the 1. control block the SP selection from digital input (**ConF./Cnt1/SEt[2]=1**) as you do not get the **PAr1./ mSP*** (setpoint chosen from digital input). It is obvious that that you ought to configure the setpoint selection first.

The flowcharts and diagrams help you to find the right configuration sequence:

Configuration navigation diagram
Control block configuration
Input block diagram
ALARM function diagram

18.2. Set-point control

The programmer does not operate in SP program control mode. The contoller is in OFF state. Write down the properties of the control loop before the configuration as it is below:

1. one control loop with one PID, one relay for control, heating
2. selection of SP by code switch from four values, **mSP1=140, mSP2=142, mSP3=144, mSP4=146**
3. autotune from the front panel
4. B type derivative (rate)
5. minimal SP **SPLo=139**
6. maximal SP **SPHi=150**
7. actuator cycle time **Yt.=15**, if it was smaller would set bigger to elongate the relay lifetime
8. minimal actuator value **YLo=3.0,** for elimination the short on-off switch time
9. maximal actuator value **YLo=97.0,** for elimination the short off-on switch time
10. relay action at PV=146 with 1 hysteresis
11. outer KTY type cold junction
12. three lead PT100 input with 0.1 resolution, Celsius degree
13. linear output for PV, **PvLo=0**, **PvHi=200**, 0 … 20 mA

The configuration: The controller can be set to 1 or 0.1 resolution. The resolution determines all of the parameter values. Therefore 100 changes to 10.0 after configuring 0.1 resolution. So we propose that begin configuration in **CAL.** page. If you must change the configuration of resolution, do not forget to check an correct the altered values. The effects of changed resolution can be seen in after **CAL./In*** and **CAL./In'*** tables.

CAL./CJ = 00000000
CAL./In1/dEF = 01110000
CAL./Lin1/dEF = 0--00000
CAL./In1/LiLo = 0
CAL./In1/LiHi = 200
ConF./Cnt1/dEF = 00000001
ConF./Cnt1/SEt = 00000100
ConF./Cnt1/dEcL = 00100000
ConF./Cnt1/OPt = 00001000
SPLo = 139.0
SPHi = 150.0
Yt = 15
YLo = 3.0
YHi = 97.0
ConF./ALr5/dEF = 00100000
AL./ALr5/ALSP = 146.0
AL./ALr5/ALhY = 1.0
PAr1./mSP/mSP1 = 140
PAr1./mSP/mSP2 = 142
PAr1./mSP/mSP3 = 144
PAr1./mSP/mSP4 = 146

18.3. Program control

The SP for the program control is generated by the programmer. It is detailed in section "Using SP Programmer". The programmer is global; all of the channels get SP from here. Let see some example about the modes of program control:

- every channel (1 … 4) gets the same SP, which can be shifted to the first. The shift value must be written in **PAr*./SP** e.g. in the second cannel. If in **PAr2./SP** = 10, then the setpoint of the second channel will be more by 10 than the first.
- four types program operating mode by the settings in **Conf./PrG/dEF[10]**.
- any of the channels can be removed from program mode. You can configure the second channel by ④→ (see the flowchart Control block configuration). After it, the second channel will work as an SP controller using the selected SP (by digital inputs).
- the SP of the programmer can be modified by the ③→ too. E.g. let's add in the third channel to the value of **PAr3./m-SP/mSP1=0** the value of another device (could be a controller) which is accepted at **In6** input. After this configuration the third channel will work with its SP + the remote SP.

As seen above, there are indefinite possibilities in the KD9 controller. Therefore, you can solve any program control tasks.

The next example shows a complicated, program controlled process.

A three-zoned drying chamber would be heated by a program so that the atmosphere would change by three types of gases and the pressure must be constant. (The example shows the simplest solution.)

- Configuration of the cahnnels:

1. channel	**Conf./Cnt1/dEF = 00000001**	**SPLo = 0**	**SPHi = 200**
2. channel	**Conf./Cnt2/dEF = 00010001**	**SPLo = 0**	**SPHi = 200**
3. channel	**Conf./Cnt3/dEF = 00100001**	**SPLo = 0**	**SPHi = 200**
4. channel	**Conf./Cnt4/dEF = 00110001**	**SPLo = 0**	**SPHi = 9999**
4. channel	**Conf./Cnt4/SEt = 00000100**		
	ConF./SYSt/mmi2 = 0000110	(from digital input SP)	
	Stnd./dInP = 00000001		
	PAr4./m-SP/mSP1 = 45 (45 mbar SP for pressure of chamber)		
1, 2, 3 channel	**YLo = 3 YHi = 97**		

- Configuration of inputs:

1. channel	**CAL./In1/dEF = 00100000**	
2. channel	**CAL./In2/dEF = 00100000**	
3. channel	**CAL./In3/dEF = 00100000**	
4. channel	**CAL./In4/dEF = 01011000**	**PvLo = 0.0 PvHi = 100.0(0-100 mbar sensor)**
4. channel	**CAL./Lin1/dEF = 0--01111**	**LiLo = 0 LiHi = 100 (for 0-20 mA input type valve)**

- Configuration of programmer

ConF./PrG/dEF = 00000010
ConF./PrG/SEt = 00000010
ConF./PrG/OPt = 00000001

- Configuration of ALARM-s:

  valve actuator for gas "A" ALr5 ↔ R5 **ConF./Alr5/dEF10010000**
  valve actuator for gas "B" ALr6 ↔ R6 **ConF./Alr6/dEF10011001**
  valve actuator for gas "C" ALr7 ↔ R7 **ConF./Alr7/dEF10010010**

- events in program:

  for gas "A" : **EvnL = 00000001**
  for gas "B" : **EvnL = 00000010**
  for gas "C" : **EvnL = 00000100**

The program

Program segment	timE	rAmP	SOAK	FLAG instruction	FLAG value	SP1	EvnL	EvnH
00	0:30					50	00000000	
01	1:00					50	00000000	
02	1:30					90	00000001	
03	2:00					90	00000010	
04	3:00					90	00000100	
05	2:00					40	00000001	
06				End			00000000	

18.4. Cascade control

Systems with very big lag time can be controlled too slowly. The workpiece may be in a retort in the furnace. The retort is closed round by the heated space. The sensor of this space sends its value to the input with lag time, which is in accordance with the time constant of the system. The controller sets the actuator by the input and the SP. The actuator alters the energy in case of e.g. 33.3% therefore ceases the whole energy of the furnace to one third.

The furnace space will be heated up fast to the SP without oscillating. The retort heats up slowly and as it approximates the SP, it will be much slowlier. The heating process needs indefinite time theoretically, but it is quite the same in practice.

Therefore, the technologist sets the SP higher. The difference called **ΔT** will be determined by the properties of technology, furnace, workpiece, temperature and many other things. Therefore, this **ΔT** is good only for one process to which was experimentally determined.

The KD9 automate this process by cascade control. The cascade control consists of embedded control loops. Every control loop has own sensor. The inner loop is the MASTER, the outer the SLAVE. The master is the "workpiece" by this example and shows the demand on energy to the SLAVE input. The output of the MASTER (Y) features the energy state of the workpiece and it will be accepted at the **In'8** auxiliary input. The signal of the auxiliary input will be transformed into **ΔT** and will be added to the SP of SLAVE. This correction additive depends only on the energy state of the MASTER and vanishes as the MASTER approaches the SLAVE. There is a remaining value in steady state because of the gradient that exists in practical systems. You can set this value in the configuration. Practically you can give the maximal and minimal **ΔT**. The maximal value will be valid at the beginning of the process and the minimal at the end.

The process called **ΔT cascade control**

The next example shows a ΔT cascade control:

The sensors of MASTER and SLAVE are PT100 RTD-s. (Settings that identically equal 0, are not written down.)

MASTER: first channel, **In1** input, SP control, there is not relay output, INP1 pins for the input.

ConF./Cnt1 = identically 0 there is not relay output

SPLo = 0
SpHi = 300
YLo = 0
YHi = 100.0

PAr1./Pid1/Gain = 5.0 proportional band: 20
PAr1./Pid1/Int = 0
PAr1./Pid1/dEr = 0

CAL./In1/dEF = 00100000 Pt100

SLAVE: second channel, **In2** input, SP control, one relay output, INP2 pins for the sensor.

ConF./Cnt2/dEF = 00010001 Pt100, control with 1 relay
ConF./Cnt2/SEt = 00001011 MASTER setpoint (**SP1**) adding to **In'8**, while (own **PAr2./SP** = 0),
SpLo = 0
SpHi = 300
Yt = 0
YLo = 0
YHi = 100.0

PAr2./Pid1 tuned PID parameters

CAL./In1/dEF = 00100000 Pt100

Auxiliary input

CAL./In'8/dEF = 00001100 the output value of the MASTER **Stnd./Y1** will be the input of **In'8**
InLo = 0 the lower value of input is 0, because here the MASTER has not energy demand
InHi = 100 the upper value of input is 100, because here the MASTER has maximal energy demand
PvLo = 4 **ΔT = 4** because the MASTER needs so much extra energy for holding the temperature (PV=SP)
PvHi = 20 **ΔT = 20** the error of the SLAVE (PV-SP) will be grown with this value to force heating of the workpiece

As you can see, the **ΔT** is proportional to the output state of the MASTER. It is the biggest at the lower point of the proportional band and the smallest at the upper point. The proportional band = 100/GAin and its upper point is at the SP. The values in the example are: in MASTER Gain=5, proportional band=100/5=20, therefore if in the MASTER SP=100 and PV=80, the SLAVE has an SP=120 value. When the PV in MASTER is 100, than in the SLAVE will be SP=104.

18.5. Heat-cool control

Some systems working at temperature near its environmental temperature, or have high-effective heat isolation need cooling.

There are many methods for HEAT-COOL control, but the common properties are:

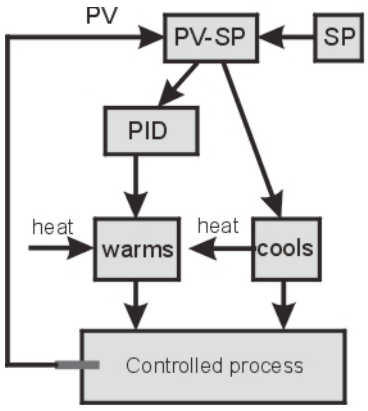
1. The controlled system has 1 sensor.
2. Two actuators control the heat flow, a heater and a cooler.
3. The controlled system has only one setpoint..

The algorithm seems very simple, but appearances can be deceptive. One primitive solution may be: when it is cool heat and when is hot cool please.

In systems with big lag time the heat control always causes oscillation. If it is true, it must be true for cooling too. You can see that is the problem. The cooling side of the system has very different properties e.g. cooling velocity vs. heating velocity, heat flow direction through the isolation, heat transfer mode et. Summarized the system has two sets of properties, therefore has two optimal parameter sets. It was proved by experiments that the P (proportional band) at cooling side differs only from at heating side value. I (reset) and D (rate) are very similar on both sides.

The problem written above can be solved by to ways:

I. You can conduct out a big heat quantity, fast arisen from technology, by an ALARM relay. Such method can be seen in Figure 20. The control can be solved by setting **PAr*./Pid*/cGn = 0.0**. In this case the cooler will work in **on-OFF** mode.



PV temperature of the process
SP setpoint (ALARM)
PV-SP error
PID PID parameters on heat side

Figure 20

It is well seen in the Figure 20, that an on-OFF and a PID control changes each other. Because an on-OF and a PID control alter in proximity of the setpoint, every change generates a disturbance. Therefore, this method is good only for compensation of big and fast heat quantity changes.

II. To eliminate the disturbances there are two PID loops in a HEAT-COOL channel. The COOL loop has different **cGn** gain and **cYt** cycle time. You can practically multiply the heater values with a **c** constant. The **GAin** could be set by autotune and to fine-tune afterwards. The Figure 21 shows the two control loop.

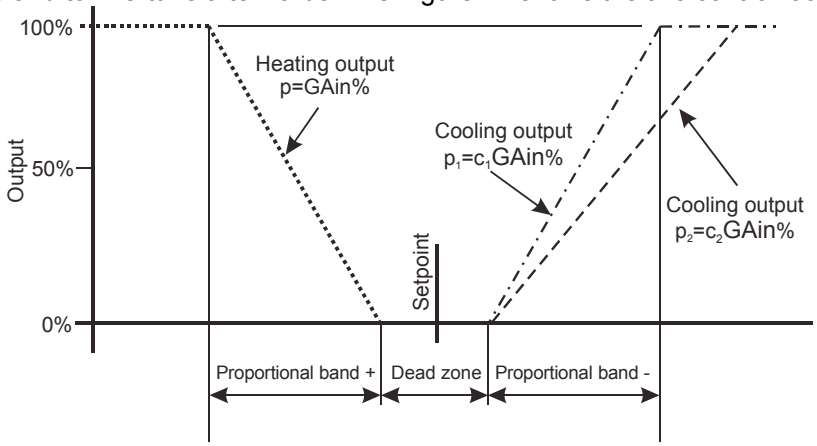
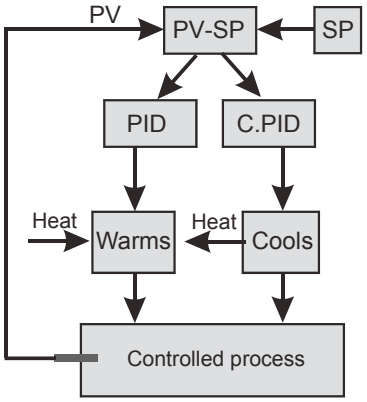


Figure 21

The actuator is 100% from left end of the proportional band of heating and near the SP is 0%. The setpoint is not a value but a range, which is called **dead zone.** You must estimate the dead zone by the system properties. The dead zone may be positive, zero and negative. The control quality depends on this value. If it is positive, either of the actuator does not work, while PV is in the dead zone range. At zero, either of the actuator will has to work by the smallest change of PV. If it is negative, either of the actuator will work together in the dead zone by the PID algorithm. The energy consumption is growing because of the cooler must conduct out the heat quantity when the heater and cooler work together. It may be expensive, but it is a very accurate method.

The proportional band of the COOL control loop is on the right of the SP (dead zone). The COOL control loop works similarly to the HEAT control loop. Only the proportional band of the cooler differs from the heater. The gain of the cooler is multiplied by a constant so: **cGn,** where **c>0**. The **c** determines the lower limit of the proportional band.

The proportional band range decreases by increasing gain. I you give a big gain (more than 50) the cooler will work in on-OFF mode, if an ALARM worked for it. It is more advantageous than using an ALARM, because you need not adjust the SP of the ALARM.



PV	temperature of the process
SP	setpoint (ALARM)
PV-SP	error
PID	PID parameters on heat side
C.PID	PID parameters on cool side

Figure 22

The Figure 22 shows, that there are two equal PID controllers in the two control loops. The loops have common SP and sensor. The controller operates the two actuators by common error signal.

The parameters can be autotuned but the **c** and the dead zone must be estimated. It is proposed to adjust these parameters experimentally.

In on-OFF mode, there is a special hysteresis for HEAT-COOL control. The two sides may be set independently. The both hysteresis's are symmetrical and their center point are at the limits of the dead zone. The settings and operations can be seen in Figure 23.

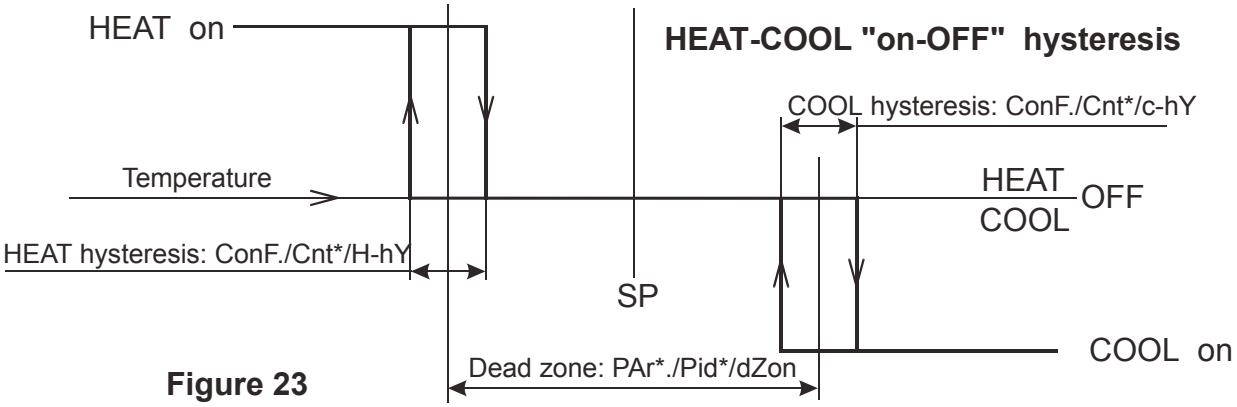


Figure 23

There are two HEAT-COOL configurations below:

HEAT-COOL CONTROL with two PID loops

ConF./Cnt1/dEF = 00000010 HEAT-COOL control. Occupies the R1 and R5 relays
- **SPLo = 0.0**
- **SPHi = 300.0**
- **Yt. = 10** cycling time of heater
- **YLo = 3.0** minimal output of heater
- **YHi = 97.0** maximal output of heater
- **cYt = 15** cycling time of cooler
- **cYLo = 3.0** minimal output of cooler
- **cYHi = 97.0** maximal output of cooler

CAL./In1/dEF = 01100000 Pt100, in 0.1 resolution

PAr1./Pid1
- **Gain = 4.2** gain of heater
- **Int = 86** reset
- **dEr = 22** rate
- **dZon = 4.0** dead zone
- **cGn = 5.5** gain of cooler

HEAT-COOL "on-OFF" control ***(see in Figure 20)***

ConF./Cnt1/dEF = 00000010 HEAT-COOL control. Occupies the R1 and R5 relays
- **SPLo = 0.0**
- **SPHi = 300.0**
- **Yt. = 10** cycling time of heater
- **YLo = 0.0** minimal output of heater
- **YHi = 100.0** maximal output of heater
- **H-hY = 1** hysteresis on the heater side
- **cYt = 15** cycling time of cooler
- **cYLo = 0.0** minimal output of cooler
- **cYHi = 100.0** maximal output of cooler
- **c-hY = 4** hysteresis on the cooler side

CAL./In1/dEF = 01100000 Pt100, in 0.1 resolution

PAr1./Pid1
- **Gain = 0.0** gain of heater
- **Int = 0** reset
- **dEr = 0** rate
- **dZon = 6.0** dead zone
- **cGn = 0.0** gain of cooler

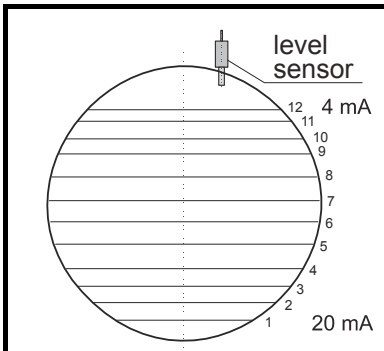
18.6. Level control

There are a lot of methods for level measure and control. One of the simplest is to hold a level by a fixed sensor signal. You can close the inlet valve when the set level is reached. The controller can work with hysteresis.

You can solve complex tasks with a level measuring sensor (acoustic, radar, ultra sound, etc.). The controller gets the output signal of the level measuring sensor (normally in 0/4 … 20 mA) some on its input. The output of the controller will operate the actuators (e.g. the valves).

The next example is a control of volume of a liquid, filling, pouring and measuring. This system is able to inlet different liquids with different volumes, therefore you can make different mixtures by technological receipts. This property is especially useful in food and chemical industries.

You can see the volume-calibration data of a horizontal tank. You can save up to seven calibration data in **utb*.** page.



Ordinal: **n**	*Level*. Row: co n/coin	*Volume*. Column: co n/coou
1	0,136	0,296
2	0,266	0,796
3	0,387	1,364
4	0,518	2,064
5	0,699	3,128
6	0,865	4,162
7	1,020	5,158
8	1,196	6,274
9	1,367	7,322
10	1,478	7,964
11	1,609	8,666
12	1,694	9,080

The data are monotone growing!

Let's change the volume of liquid in the tank. You must write the calibrated data into the **utb1.** page. Using this tabulated function you can transform the signal of the level-measuring sensor into volume. The function linearizes the input between 0.136 … 1.694, according to table above. So the when the level signal is 12 mA then the volume will be 9,08/2 = 4.54 m^3. After configuration a control loop and a setpoint of 5 is given, the inlet or the outlet valve will be active until the volume will not be 5 m^3. You must use for this purpose the HEAT-COOL control in on-OFF mode, where HEAT relay operates the inlet and the COOL the outlet valve. The mode is quite understandable if we know that at temperature control the input is "mV" and output is "°C ". At level control, the input is "level" and output is "volume".

Configuration

ConF./Cnt1/dEF = 00000010 HEAT-cool control. It occupies the R1 and R5 relays. R1 → fills, R5 → empties
- **SPLo = 0.000** the minimal value for volume (out of this the controller sends an error message)
- **SPHi = 9.999** the maximal value for volume (out of this the controller sends an error message)
- **Yt. = 1**
- **YLo = 0.0**
- **YHi = 100.0**
- **H-hY = 0.002** hysteresis on the filling side
- **cYt = 1**
- **cYLo = 0.0**
- **cYHi = 100.0**
- **c-hY = 0.002** hysteresis on the emptying side

CAL./In1/dEF = 11011001 three virtual decimal, current input: 4-20 mA
CAL./In1/math = 00010000 the first customized linearization table
- **InLo = 0**
- **InHi = 0**
- **PvLo = 0.136** lower value of linearization (level measuring sensor signal is 4 mA)
- **PvHi = 1.694** upper value of linearization (level measuring sensor signal is 20 mA))

PAr1./Pid1
- **Gain = 0.0**
- **Int = 0**
- **dEr = 0**
- **dZon = 0.006** dead zone
- **cGn = 0.0**

Manual control is in **Stnd./Y1**, you can open and close the valves by setting the value on the display. -100.0 empties, 0.000 valves close, 100.0 fills. The volume can be see on the upper red display.

18.7. Relative humidity control

The RH can be measured and controlled by capacitive sensor. The capacitance changes with the humidity of the dielectric. Another method is for RH measuring the use wet/dry thermometers. Every method can be used in clean environment and by careful supervision. The sensor loses its accuracy if it is contaminated. Where the technology contaminates the sensor there may be useful to apply wet/dry thermometers. The KD9 can control six RH if they have linear (current or voltage) output.

The wet/dry thermometers method uses two inputs. The controller subtracts the wet bulb from the dry bulb so makes the psychometric difference. Then it counts the RH by the custom linearization table saved in memory. The system can be controlled by this result.

The next example shows a possible configuration. The application could be solved more simply, but the example would like to show you the great abilities of the KD9.

This table was written for control in the **mtr1.** page.

Column index [**m**]→		1	2	3	4	5	6	7	8	9	A
Row index [n]	**dry T_{dr}** °C	**dry-wet ΔT = T_{dr}-T_w** °C									
		1	2	3	4	5	6	7	8	9	10
1	10	88	77	66	55	44	34	24	15	6	0
2	12	89	78	68	58	48	39	29	21	12	0
3	14	90	79	70	60	51	42	34	26	18	10
4	16	90	81	71	63	54	46	38	30	23	15
5	18	91	82	73	65	57	49	41	34	27	20
6	20	91	83	74	66	59	51	44	37	31	24
7	22	92	83	76	68	61	54	47	40	34	28
8	24	92	84	77	69	62	56	49	43	37	31
9	26	92	85	78	71	64	58	51	46	40	34
A	28	93	85	78	72	65	59	53	48	42	37
b	30	93	86	79	73	67	61	55	50	44	39

T_{dr} data: **mtr1./r.^{coo}/rov[n]**, where **n** is the row index of the matrix (1 …9, A, b). E.g. **mtr1./r.^{coo}/rov8 = 24**
ΔT data: **mtr1./r.^{coo}/CoL[m]**, where **m** is the column index of the matrix (1 …9, A). E.G **mtr1./r.^{coo}/CoLA = 10**
When the dry bulb is 24ºC and the wet bulb is 14ºC then to the difference (ΔT = 10ºC) belongs 31% RH.
So **8.rov/A.CoL = 31** by the table.

Note. This table is the same as the other linearization tables, which are used to the sensors. The linearization tables were calibrated by International Organizations and the producers do their best to approximate these data with their sensors. The error of a sensor is its deviation from the table. The controller contains many tables in its memory. When you select a sensor, the controller automatically attaches it to a table. There are some special sensors, which have not standardized tables. The Kd9 can store seven functions (in table) with one variable and two functions (in table) with two variables in the nonvolatile memory.
The example shows a RH control however the KD9 can control with the method any other similar systems (Carbon potential, uneven cross sectioned tank level control, chemical processes with two variables etc.)

Wiring	Dry bulb:	INP1 → 15 16 pins,	2-wire Pt100
	Wet bulb:	INP2 → 14 16 pins,	2-wire Pt100

The configuration of the controller

The control blocks (number 1, 2, 3 and 4) can be called successively by pushing the key)

Relative humidity **Conf./Cnt1/dEF = 01110001**, controls with one relay, appears on the red display (number **1**)
SPHi = 100

Dry bulb **Conf./Cnt2/dEF = 00000000**, on the red display (number **2**)
SPLo = 000
SPHi = 100

Wet bulb **Conf./Cnt3/dEF = 00010000**, on the red display (number **3**)
SPLo = 000
SPHi = 100

ΔT (psychrometric difference) **Conf./Cnt4/dEF = 01100000,** on the red display (number **4**)
SPLo = 000
SPHi = 100

Calibration

1. input **CAL./In1/dEF = 01100000**, one decimal, Pt100
FILt = 10001000

2. input **CAL./In2/dEF = 01100000**, one decimal, Pt100
FILt = 10001000

7. input **CAL./In'7/dEF = 00000001**, **Stnd./In1** dry bulb value
mAth = 00001010 subtracts the wet bulb value (**Stnd./In2**)

8. input **CAL./In'8/dEF = 00000001**, the wet bulb signal, **Stnd./In1** the row in the header of the table
FILt = 00000111
mtrll = 00000111 number of table **mtrll[4] = 0**, "column" of table **mtrll[3210] = 0111**

18.8. Motorized valve control

The motorized valve control rotates the motor with two independent relays. The motor has three states: rotates clockwise, stops, rotates counterclockwise. The quality of control determines the lifetime of the motorized valve. Fewer movement, longer lifetime. The control accuracy need frequent actuations. Due the two contrary demands, we must find the optimal solution. The software of the KD9 solves this problem.

The quality of control depends on settings. The next data must be found by experiments:

1. Determine PID parameter values by autotune and adjust them experimentally.
2. For determining the traversing time measure the rotation time from opened to closed state. The fastest method is to use manual control. Write this value in menu item **Yt.** in seconds. The smallest valid actuator value must be written in **Yd'** menu item. Recommended value: **Yd'**=1 … 5. This value minimizes keep jerking. The motorized valve can control in SP or SP program mode. Every data of control can be changed in on or OFF mode.

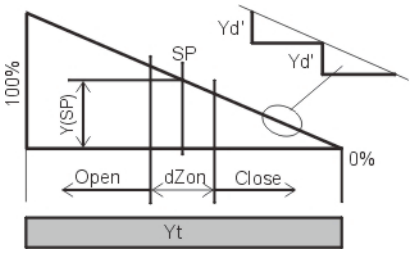
Meanings of data

Yt. is the time during which the valve opens from closed state caused the jump Y=0 → Y=100 (traversing time).

Y(SP) actuator value at steady state.

dZon dead zone in which the actuator does not rotate the motor.

Yd' the smallest value of the actuator. Under this value the actuator does not rotate the motor



The system properties need two type of wiring because the dangerous state varies by the technology. In the brooder house the valve must be held in the last sate, but in steam-boiler system the valve must be closed when a breakdown occurs. The wiring can be seen in the Figure 24

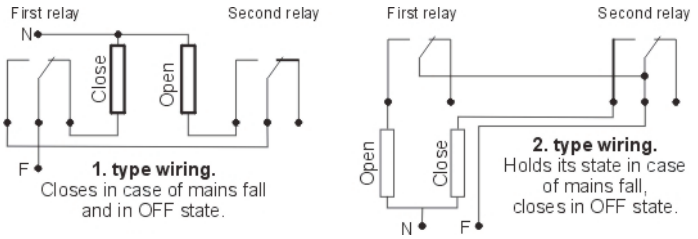


Figure 24

The configuration:

ConF./Cnt*/dEF[10] = 11
ConF./Cnt*/Yt.
ConF./Cnt*/Yd'

The control mode can be configured like the other modes.

19. MODBUS register map

(from version 1.012)

The accessibility of the menu items and registers can alter by the build up and the configuration of the controller. Study the manual if some register does not answer or their values are zero.

The KD9 controller communicates by Standard Modbus protocol with other devices. Download **www.modbus.org**. The communication interface of the controller can simply connected to any SCADA and visualization programs.

NOTE, the numbers of the register addresses in visualization programs begins with **1** so increase this addresses by **1**.

All of the register numbers are in hexadecimal form. The MENU is big, compound and repetitive so addresses are by the their rout written in the table. Count the valid address by algebraic operation from address part of the route.

Example: the counting of address of **AL./ALrC/AlHy**:

100 AL :+6*400 AlrC: +C0 ALHY: +1 The result: **100+6*400+C0+1=19C1**, where * is a multiplication.

The special registers which cannot achieve from MENU and the program area, can be found from **0** to **100**.

Address	Function	Register number	Meaning
20	03,04,06	1	States of controller **.7**: Systemr **1**:ON / **0**:OFF; **.5**: **ConF** page disable sign; **.4**: **HOLD** state sign
28	03,04	1	Read digital input
30	03,04	2	Programmer state, program time **<S; x_3*2^{16}+ x_2*2^8+ x_1>** **X**: spent time in a program step (segment) in s. **S**: **0**:?, not decodable commands, **1**:SOAK, **2**:run, **3**:run-, **4**:timE, **5**:FLAG, **6**:FrEE, **7**:SLAvE
38	03,04,06	1	Program start and editing program and step (segment) number <Hi byte: program number><Low byte: step (segment) number > **NOTE** *Whe editing program the rewrite may cause faulty program alteration!*
40	03,04	1	State of ALARM relays
48	03,04	1	**Stnd./dInP/(rEGH, rEGL**) always readable and writable
80	03,04,10	2	In1 = (double register / 40) (hexadecimal value)
82	03,04,10	2	In2 = (double register / 40) (hexadecimal value)
84	03,04,10	2	In3 = (double register / 40) (hexadecimal value)
86	03,04,10	2	In4 = (double register / 40) (hexadecimal value)
88	03,04,10	2	In5 = (double register / 40) (hexadecimal value)
8A	03,04,10	2	In6 = (double register / 40) (hexadecimal value)
8C	03,04,10	2	In'7 = (double register / 40) (hexadecimal value)
8E	03,04,10	2	In'8 = (double register / 40) (hexadecimal value)
C0	03,04,10	2	SP1 = (double register / 40) (hexadecimal value)
C2	03,04,10	2	SP2 = (double register / 40) (hexadecimal value)
C4	03,04,10	2	SP3 = (double register / 40) (hexadecimal value))
C8	03,04,10	2	SP4 = (double register / 40) (hexadecimal value)

The SP programmer can operate in more working method, in multy programmer mode the register map changes!

Sequencer and simple SP programmer.

The registers of SP programmer are readable by 03 and 04 functions; they are writable by 06 function but at once only one.

Program steps (segments) from address	$8000 continuously follow each other.
Time and type of program step are accessible at	$8000 + 3*(100* (program serial number)+(step serial number) address.
SP value of program step is accessible at	$8000 + 3*(100* (program serial number)+(step serial number)+1 address.
The event code of program step is accessible at	$8000 + 3*(100* (program serial number)+(step serial number)+2 address.

Relation between type and time:
RampX : 0…5999, if value of register is in this range then the program step is RampX type.
RampR (6000+ 1999) + (-1999…9999) if value of register is in this range {6000…17998}, then the program step is ramp type.
SOAK (17998+(0+..5999) if value of register is in this range { 17998…23997}, then the program step is soak type
Above 23997 register values are the serial numbers of FLAG-s. See the detailed description of FLAG-s in the KD9 manual.

The KD9 can save only 50 SP program from "multi programmer type" **(0…49)**.
The register map formulas alter as follows:

Time and type of program step are accessible at	$8000 + 6*(100* (program serial number)+(step serial number) address.
SP value of program step is accessible at	$8000 + 6*(100* (program serial number)+(step serial number)+1 address.
The event code of program step is accessible at	$8000 + 6*(100* (program serial number)+(step serial number)+2 address.
SP2 value of program step is accessible at	$8000 + 6*(100* (program serial number)+(step serial number)+3 address.
SP3 value of program step is accessible at	$8000 + 6*(100* (program serial number)+(step serial number)+4 address.
SP4 value of program step is accessible at	$8000 + 6*(100* (program serial number)+(step serial number)+5 address.

Relation between type and time does not alter, but giving the RampX and SOAK has not any meaning.
It is similar at setting a sequencer, the RampX and SOAK types have not any meaning.

The registers of MENU are readable by 03 and 04 functions; they are writable by 06 function but at once only one.

100	+		+		+
Stnd	0*400				
		vers	00		
		Load	10		
		In1	20		
		In2	30		
		In3	40		
		In4	50		
		In5	60		
		In6	70		
		In'7	80		
		In'8	90		
		CJ	A0		
		Y1	B0		
		Y2	C0		
		Y3	D0		
		Y4	E0		
		Al.rb	F0		
				AlbL	0
				ALbH	1
				RLbL	2
				RLbH	3
		dinP	100		
				dInp	0
				Cnt1	1
				Cnt2	2
				Cnt3	3
				Cnt4	4
				Cnt5	5
				Cnt6	6
				Cnt7	7
				Cnt8	8
				rEGL	9
				rEGH	A
		P.Edt	110		
				Src.S	0
				Src.E	1
				DSt.S	2
				OPCd	3
		PrG	120		
				PrFL	0
				SttS	1
				REG1	3
				REG2	4
				REG3	5
				REG4	6
				EvnL	7
				EvnH	8
				PrEt	9
		Sont	130		
				H/m	0
				day	1
		rtc	140		
				sec	0
				H/m	1
				day	2
				mont	3
				year	4
		ucLb	150		
				WA^m	0
				uGn1	1
				uGn2	2
				uGn3	3
				uGn4	4
				uGn5	5
				uGn6	6
100	+		+		+
StAt	1*400	rset	00		
		dE1	10		
		dE2	20		
		dE3	30		
		dE4	40		
				dELo	0
				dEHi	1
100	+		+		+

Par1	2*400				
Par2	3*400				
Par3	4*400				
Par4	5*400				
		SP	00		
		m-SP	10		
				mSP1	0
				mSP2	1
				mSP3	2
				MSP4	3
		Pid1	20		
		Pid2	30		
		Pid3	40		
		Pid4	50		
		Pid5	60		
		Pid6	70		
		Pid7	80		
		Pid8	90		
		Pid9	A0		
		PidA	B0		
		Pidb	C0		
		Pidc	D0		
		Pidd	E0		
		PidE	F0		
		PidF	100		
		PidG	110		
				Gain	0
				Int	1
				Der	2
				Mres	3
				dzon	4
				CGn	5
				Yt	6
				cYt	7
100	+		+		+
AL	6*400				
		Al.rb	00		
				AlbL	0
				ALbH	1
				RLbL	2
				RLbH	3
		ALr1	10		
		ALr2	20		
		ALr3	30		
		ALr4	40		
		ALr5	50		
		ALr6	60		
		ALr7	70		
		ALr8	80		
		ALr9	90		
		ALrA	A0		
		ALrb	B0		
		ALrC	C0		
		ALrd	D0		
		ALrE	E0		
		ALrF	F0		
		ALrG	100		
				ALSP	0
				ALhy	1
100	+		+		+
Conf	8*400				
		Syst	00		
				Mmi1	0
				Mmi2	1
				Mmi3	2
				Mmi4	3
				Mmi5	4
				mmi6	5
				DFLt	6
		PrG	10		
				DEF	0
				SEt	1
				DECL	2
				OPT	3
				SSP1	4

				SSP2	5
				SSP3	6
				SSP4	7
				EvnL	8
				EvnH	9
		Cnt1	20		
		Cnt2	30		
		Cnt3	40		
		Cnt4	50		
				dEF	0
				SEt	1
				DECL	2
				OPT	3
				SPLo	4
				SPHi	5
				Yt.	6
				Yd'	7
				YLo	8
				YHi	9
				H hY	A
				cYt	B
				cYLo	C
				cYHi	D
				C HY	E
				2nIn	F
		Stat	60		
		Alr1	70		
		Alr2	80		
		Alr3	90		
		Alr4	A0		
		Alr5	B0		
		Alr6	C0		
		Alr7	D0		
		Alr8	E0		
		Alr9	F0		
		AlrA	100		
		Alrb	110		
		AlrC	120		
		Alrd	130		
		AlrE	140		
		AlrF	150		
		AlrG	160		
				DEF	0
				SET	1
				LGA1	2
				LGA2	3
				LGE1	4
				LGE2	5
				Decl	6
				Opt	7
				Soft	8
				ALdt	9
				AHdt	A
				ALSP	B
				ALHY	C
		rtc	170		
				rtc	0
				sec	1
				H/m	2
				day	3
				mont	4
				year	5
		Com1	180		
		Com2	190		
				Def	0
				Set	1
				Addr	2
				OSP1	3
100	+		+		+
CaL	A*400				
		CJ	00		
				Def	0
				SHFT	1
		In 1	10		
		In 2	20		

		In 3	30		
		In 4	40		
		In 5	50		
		In 6	60		
				DEF	0
				Unit	1
				Filt	2
				DECL	3
				Math	4
				Shft	5
				Inlo	6
				inhi	7
				PVLO	8
				PVHi	9
				uGn	A
		In'7	70		
		In'8	80		
				DEF	0
				Unit	1
				Filt	2
				DECL	3
				math	4
				shft	5
				inlo	6
				inhi	7
				PVLO	8
				PVHi	9
				Mtrx	A
		Lin1	90		
		Lin2	A0		
		Lin3	B0		
		Lin4	C0		
				def	0
				LiLO	1
				LiHi	2
		dCnt	D0		
				Ctr1	0
				Ctr2	1
				Ctr3	2
				Ctr4	3
				Ctr5	4
				Ctr6	5
				Ctr7	6
				Ctr8	7
		File	E0		
				Def	0
				Set	1
				Reg1	2
				Reg2	3
				OPt	4
100	+		+		+
Prnt	B*400				
		def	00		
		set	10		
		decl	20		
		opt	30		
		ReG1	40		
		ReG2	50		
		ReG3	60		
		ReG4	70		
		ReG5	80		
		ReG6	90		
		ReG7	A0		
		ReG8	B0		
		ReG9	C0		
		ReGA	D0		
		ReGB	E0		
		ReGC	F0		
				def	0
				PrLo	1

				PrHi	2
100	+		+		+
Utb1	D*400				
Utb2	E*400				
Utb3	F*400				
Utb4	10*400				
Utb5	11*400				
Utb6	12*400				
Utb7	13*400				
		co 1	00		
		co 2	10		
		co 3	20		
		co 4	30		
		co 5	40		
		co 6	50		
		co 7	60		
		co 8	70		
		co 9	80		
		co10	90		
		co11	A0		
		co12	B0		
		co13	C0		
		co14	D0		
		co15	E0		
		co16	F0		
		co17	100		
		co18	110		
		co19	120		
		co20	130		
		co20	140		
		co22	150		
		co23	160		
		co24	170		
		co25	180		
		co26	190		
		co27	1A0		
		co28	1B0		
		co29	1C0		
		co30	1D0		
		co31	1E0		
		co32	1F0		
				coin	0
				cou	1
100	+		+		+
Mtr1	14*400				
Mtr2	15*400				
Mtr3	16*400				
		c.coo	00		
				CoL.1.	0
				CoL.2.	1
				CoL.3.	2
				CoL.4.	3
				CoL.5.	4
				CoL.6.	5
				CoL.7.	6
				CoL.8.	7
				CoL.9.	8
				CoL.A.	9
				CoL.B.	A
				CoL.C.	B
				CoL.D.	C
				CoL.E.	D
				CoL.F.	E
				CoL.G.	F
		r.coo	20		
				rov.1.	0
				rov.2.	1
				rov.3.	2
				rov.4.	3
				rov.5.	4

				rov.6.	5
				rov.7.	6
				rov.8.	7
				rov.9.	8
				rov.A.	9
				rov.B.	A
				rov.C.	B
				rov.D.	C
				rov.E.	D
				rov.F.	E
				rov.G.	F
		1.col	40		
		2.col	50		
		3.col	60		
		4.col	70		
		5.col	80		
		6.col	90		
		7.col	A0		
		8.col	B0		
		9.col	C0		
		A.col	D0		
		B.col	E0		
		C.col	F0		
		D.col	100		
		E.col	110		
		F.col	120		
		G.col	130		
				1.rov	0
				2.rov	1
				3.rov	2
				4.rov	3
				5.rov	4
				6.rov	5
				7.rov	6
				8.rov	7
				9.rov	8
				A.rov	9
				B.rov	A
				C.rov	B
				D.rov	C
				E.rov	D
				F.rov	E
				G.rov	F
100	+		+		+
PASS	18*400				

20. Technical specification

Control functions

Control loops:	max. 4 PID + 2 on/OFF
Control modes:	SP, cascade, SP programmer
Special applications:	multi channel, MASTER-SLAVE, cascade, ratio , override, motorized valve with or without feedback, HEAT-COOL, carbon potential, dry/wet bulb humidity, mean value calculation, math functions, hot swap, etc.

Inputs

Anlogue inputs (PV)

Number of sensors:	
2 wire	max. 6
TC	max. 5 + cold junction
3 wire	max 3.
Accuracy:	±0,1%
Ranges:	mV, mA, V
Sort of TC types:	15
Sort of RTD types:	13
Sort of Thermistor:	1
Cold junction:	outer (KTY, or PT100), or fixed 0°C, or fixed 25°C

Linear inputs

Current:	0/4 - 20 mA (10 Ohm shunt is in accessory bag)
Voltage:	0 - 50 mV, 0 - 100 mV, 0 - 200 mV 4 - 200 mV 0/0,2 - 1 V 0/0,4 - 2 V 0/1 - 5 V *0/2 - 10V* (*if ordered*)
Potentiometer:	0 - 500 Ohm 0 - 5 KOhm

Digital inputs: 7 pcs (from no-voltage contact)

Outputs

Relay:	5 pcs Form C, or 11 pcs Form A, or 11 pcs SSR driver (~12 V)
Digital:	5 pcs TTL
Linear:	4 pcs (12 bit resolution) 0/4 - 20 mA 0/1 - 5 V 0/2 - 10 V
Transmitter supply	Rating, 24Vdc 100 mA (total sum of transmitters and linear outputs)

SP programmer

Total sum of segments:	10000
Number of profiles:	100
Number of profile with common time base:	4
Number of events in a segment:	16

HAGA-BASIC instructions
Events that can be activated by digital inputs.

Special properties

(Real time clock)
Timer for every ALARM function, with latches
Four counters for the SP programmer
Eight counters four the digital inputs
Statistical functions: minimax PV-SP memory

User made linearization tables:	1 variable	7 pcs
	2 variable	2 pcs
Functions from keys:	manual mode	
	hold (stops program running)	
	skip (program manual advance)	

Data acquisition on MMC (MultiMedia Card)
Built in parallel printer interface for chart recording.

Communication

MODBUS RTU	RS485 (2 wire)
	RS232 (3 wire)

MASTER-SLAVE
VISHAGA visualization software (free)
max. eight MASTER for 8 RS232 comm port in one PC
altogether max. 8x32 pcs HAGA controller in one PC.

Electrical data

Power supply:	switching type
Recommended fuses:	for controller T315 mA and for relays 5 A each.
Voltage:	85-265 VAC / 48-400 Hz, or 120-375 VDC
Consumption:	10 VA
Safety test:	Complies to MSZEN61010-1, installation category II. pollution degree 2.Isolation: Inputs and outputs are galvanic isolated, except: STR drivers, two linear outputs and RS232 interface

21. Appendix

21.1. Configurtion note

Copy these pages. Write the configuration data in the tables.
Use binary signes for EDS (electronic DIP switch) e.g. 00110011

PAr*.

PAr*./	1	2	3	4
SP				
mSP/mSP1				
mSP/mSP2				
mSP/mSP3				
mSP/mSP4				

PAr1./Pid*/	1	2	3	4	5	6	7	8	9	A	b	C	d	E	F	G
GAin																
Int																
dEr																
mrES																
dZon																
cGn																
Yt																
cYt																

PAr2./Pid*/	1	2	3	4	5	6	7	8	9	A	b	C	d	E	F	G
GAin																
Int																
dEr																
mrES																
dZon																
cGn																
Yt																
cYt																

PAr3./Pid*/	1	2	3	4	5	6	7	8	9	A	b	C	d	E	F	G
GAin																
Int																
dEr																
mrES																
dZon																
cGn																
Yt																
cYt																

PAr4./Pid*/	1	2	3	4	5	6	7	8	9	A	b	C	d	E	F	G
GAin																
Int																
dEr																
mrES																
dZon																
cGn																
Yt																
cYt																

AL. It is the same as **ConF./AL*/ALSP** and **ALhY** menu items.

ALr*	1	2	3	4	5	6	7	8	9	A	b	C	d	E	F	G
ALSP																
ALhY																

ConF.

ConF./SYSt/mmi1 =

ConF./SYSt/mmi2 =

ConF./SYSt/mmi3 =

ConF./SYSt/mmi4 =

ConF./SYSt/mmi5 =

ConF./SYSt/mmi6 =

ConF./PrG/dEF =

ConF./PrG/SET =

ConF./PrG/dEcL =

ConF./PrG/OPt =

ConF./PrG/SSP*	1	2	3	4
SSP				

ConF./PrG/EvnL =

ConF./PrG/EvnH =

ConF./Cnt*/	1	2	3	4
dEF				
SEt				
dEcL				
OPt				
SPLo				
SPHi				
Yt				
Yd'				
YLo				
YHi				
HhY				
cYt				
cYLo				
cYHi				
chY				
2nIn				

ConF./StAt =

ConF./ALr*/	1	2	3	4	5	6	7	8
dEF								
SEt								
LGE1								
LGE2								
LGA1								
LGA2								
dEcL								
OPt								
SOFt								
ALdt								
AHdt								
ALSP								
ALHY								

ConF./ALr*/	9	A	b	C	d	E	F	G
dEF								
SEt								
LGE1								
LGE2								
LGA1								
LGA2								
dEcL								
OPt								
SOFt								
ALdt								
AHdt								
ALSP								
ALHY								

ConF./Com*	1	2
dEF		
SEt		
Addr		
oSP1		

CAL./CJ/dEF =
CAL./CJ/ShFt =

CAL./In*/	1	2	3	4	5	6	'7	'8
dEF								
Unit								
FILt								
dEcL								
mAth								
SHFt								
InLo								
InHi								
PvLo								
PvHi								
uGn								
mtrll	—	—	—	—	—	—		

CAL./Lin*	1	2	3	4
dEF				
LiLo				
LiHi				

CAL./dCtr	1	2	3	4	5	6	7	8
Ctr*								

CAL./FiLE/dEF=
CAL./FiLE/Set=
CAL./FiLE/rEG1=
CAL./FiLE/rEG2=
CAL./FiLE/OPt=
Prnt./dEF=
Prnt./Set=
Prnt./dEcL=
Prnt./Opt=

Prnt./rEG*	1	2	3	4	5	6	7	8	9	10	11	12
dEF												
PrLo												
PrHi												

utb*

Drawing and note	Ordinal: **n**	*Sensor* Row: co n/coin	*Result* Column: co n/coou
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	10		
	11		
	12		

mtr*

Column index [m]→		1	2	3	4	5	6	7	8	9	A
Row index [n]	1. sensor	2.sensor									
		1	2	3	4	5	6	7	8	9	A
1											
2											
3											
4											
5											
6											
7											
8											
9											
A											
b											

1. Kd9 controller 1
1.1. Introduction 1
1.2. About this Manual 1
1.3. Installation and, wiring 2
1.4. Working principle 4
1.5. Settings using front panel keys (Handling) 5
1.6. CONFIGURATION NAVIGATION DIAGRAM 6
1.7. Configuration 8
1.8. Moving in menu 9
1.9. Menu item selection and giving parameter values 10
1.10. Setting the numeric displays (the middle and the lower) 10
1.11. Specifying linearization tables 12

2. Base (parameter) tables 14
2.1. Stnd. 14
2.2. StAt. 14
2.3. PAr 15*
2.4. AL. 15

3. ConF configuration tables 16
3.1. ConF./SYSt 16
3.2. SP Program configuration 19
3.3. Channel configuration 21
3.4. ALARM configuration 24
3.5. Realtime clock configuration 26
3.6. Communucation configuration 27

4. CAL configuration 28
4.1. Cold junction configuration 28
4.2. Input calibration 29
4.3. Mathematical input calibration 32

5. Linear output configuration 33

6. Imputs counter configuration 33

7. Data acquisition memory card configuration 34

8. Loggin and printing 35
8.1. 6 colour 32-channel hybrid data logger and chart recorder. 35
8.2. Chart recording on printer 37
8.3. The Printers 37
8.4. Priter configuration 38

9. Linearisation tables 40
9.1. utb 40*
9.2. mtr 40*

10. Flowcharts 41

11. Using SP programmer 44
11.1. The SP programmer 44
11.2. FLAG 48
11.3. The events 49

11.4. Course of writing SP program *50*
11.5. Special functions: *51*
11.5.1. AUTO-WAIT 51
11.5.2. SHADOW 51
11.5.3. The ALARM action (configuring in Conf./ALr*) 52
11.5.4. Settings of ALARM-s 52
11.5.5. System events, common events referring to the control: 52

12. ALARM timer function 52

13. PID control 54

13.1. The PID tuning *54*
13.2. Manual tuning *54*
13.3. Autotuning *54*

14. The digital inputs 56

15. The digital outputs 56

16. Error, error messages, resets 57

17. Security 58

18. Specific applications — Basic principles 59

18.1. The blocks: *59*
18.2. Set-point control *59*
18.3. Program control *60*
18.4. Cascade control *61*
18.5. Heat-cool control *62*
18.6. Level control *65*
18.7. Relative humidity control *66*
18.8. Motorized valve control *67*

19. MODBUS register map 68

20. Technical specification 71

21. Appendix 72

21.1. Configurtion note *72*